UNIVERSIDAD POLITECNICA DE VALENCIA

TESIS DOCTORAL

Departamento de Sistemas Informáticos y Computación
Programa de Reconocimiento de Formas e Inteligencia Artificial
Universidad Politécnica de Valencia

Representación, Interpretación y Aprendizaje de Flujos de Trabajo basado en Actividades para la estandarización de Vías Clínicas

Presentada por: Carlos Fernández Llatas

Dirigida por: Dr. Jose Miguel Benedí Ruiz
y Dr. Vicente Traver Salcedo

Índice general

Agradecimientos

En primer lugar me gustaría dedicar esta tesis a mi mujer, Chelo, y a mi hijo, Marcos, que son los que más han sufrido mis ausencias y se merecen este reconocimiento más que nadie. Va por vosotros!

Me gustaría dar las gracias, a mis padres, mi hermana, mis abuelos y mi familia en general, los que están y los que ya no están, que por fin, podrán dejar de preguntarme por cuando termino la tesis.

A Jose Miguel (Ponte bien, que te espero en la lectura!!) del que he aprendido como escribir una tesis y a Vicente con el que he compartido muchas horas de trabajo y al que considero, además de jefe, amigo.

A Sergio, por haberme dado la oportunidad de trabajar en el TSB, y a la postre dejarme llegar hasta donde estoy.

A Salva, al que debo mas de una cervecilla por ser mi confesor y amigo. A Juan Pablo y Andrea y David y Gema, por aguantarme y acompañarme en mi camino con una sonrisa. A Jose e Iris, Almudena, Juanmi y Maria Jesus por ser mis amigos de siempre.

A Pili, Teresa, y Aticha por su frescura y alegría. A Felipe y Amparo, por darme fuerzas desde tan lejos. A Jose Antonio, por su infinita paciencia. A Susana y lo que fue el equipo Lyra, por los buenos momentos.

A mis compañeros Juan Carlos, Eduardo, Cesar, Marta, Manolo, Paco, Nacho, Montse, Amalia y Antonio, Nieves, Alexander, de Valencia, Maria Teresa, Chiqui, Elenita (Gracias por tu apoyo en estos momentos finales de la tesis!!!) Miguel Angel, Alejandro, de Madrid, y Luis Enrique, Xtina.

Al equipo oficial de frikis: Alvarito, Alejandro, JoseFco, Juan, Carlos (Charlie), Joan, Jose (Blasco), Cesar, Serdula, Pancho, Javi, Juan(Hontanilla) y Lucas por haberme permitido entrar en su club.

A Paco: π, con esto te lo digo todo.

A tantos otros que he encontrado en mi camino, y los que me he apoyado para continuar: Carmen, Ruty y Clint, Alfredo y Bea, Pedro y Cristina, Ubaldo y Angela, Efren y Paula, Edu, Rodrigo y Almudena, ...

A Jandro (Gracias por tu magia), Iker Jimenez y todo su equipo, Fernando Alonso, Pau Gasol, Rafa Nadal y a Ben por haberme hecho este recorrido mucho más ameno.

A ellos y a todos aquellos que se cruzaron en mi camino, y se han quedado escondidos en mi memoria,

GRACIAS.

Resumen

Describir los mejores procesos para ejecutar correctamente una estrategia de una forma eficiente y con calidad no es siempre una tarea fácil. La estandarización de procesos en general y de Vías Clínicas en particular requiere de potentes herramientas de especificación e implementación que apoyen a los expertos en diseño. La utilización de modelos de Flujos de Trabajo (del inglés *Workflows*) facilita a los expertos en diseño la creación las reglas de ejecución de sus sistemas como si fueran programadores. Aún así debido a la gran mutabilidad de los procesos reales, es muy difícil conocer como los procesos se están ejecutando en la realidad. La utilización de técnicas de reconocimiento de formas pueden ayudar a los expertos en procesos a inferir, a partir de muestras de ejecución pasadas, modelos que expliquen la forma en la que estos procesos están efectivamente ejecutándose. Este paradigma es conocido como Aprendizaje de Flujos de Trabajo (del inglés *Workflow Mining*).

Los cambios de estado en los procesos de cuidado existentes en las Vías Clínicas se basan en los resultados de las acciones. Los modelos actuales de Aprendizaje de Flujos de Trabajo no recogen esta información en sus corpus. Por eso, los actuales sistemas de aprendizaje no cubren las necesidades de problemas complejos como es el caso de las Vías Clínicas.

En esta Tesis se van a estudiar los modelos de representación, interpretación y aprendizaje de Flujos de Trabajo con la intención de proponer un modelo adecuado para resolver los problemas que impiden a los diseñadores de procesos complejos, como Vías Clínicas, utilizar técnicas de Aprendizaje de Flujos de Trabajo. Para ello se va a definir un nuevo paradigma adecuado para el apoyo al diseño de Vías Clínicas, además de proporcionar herramientas para su uso. Por esto en esta Tesis se presenta además un modelo de representación de Flujos de Trabajo con una alta expresividad y legibilidad, una herramienta software capaz de ejecutar y simular Flujos de Trabajo para crear corpus de aprendizaje usando esta nueva metodología y un algoritmo de aprendizaje que sea capaz de inferir Flujos de Trabajo acordes con el problema del diseño de las Vías Clínicas.

Abstract

Description of processes that allow developing strategies with quality and in an efficient way is not an easy task. To help the process designers, powerful specification and implementation tools to standardize general processes, and particularly Clinical Guidelines, are needed. The use of Workflows help design experts in the creation of system execution rules in the way computer programmers write a program. However, the design problem is not solved because of the actual mutability of actual processes, what makes difficult to design experts know which are the current working processes. The use of Pattern Recognition techniques allows experts learning the control flow inherent to proceses and helps them knowing what is happening with the processes in reality. This methodology is known as Workflow Mining.

Clinical Pathways present state changes in their care processes that are triggered by actions results. Existing Workflow Mining methodologies do not allow inferring this information because it is not included in their corpora. Complex problems such as the Clinical Pathways case are not addressed by the current mining systems.

In this work, current Workflow Representation, Interpretation and Learning models are analyzed. This study intends to propose an adequate model to solve the problems that prevent Clinical Guidelines designers to access Workflow Mining facilities. For that, a new methodology for an adequate Clinical Pathways design as well as new tools to allows the implantation of this methodology are presented. As a result, this document presents: a new Workflow Mining methodology, a new Workflow representation model with high expressivity and legibility, a new software tool able to execute and simulate Workflows, and a new algorithm able to infer Workflows from past samples. Those results are focused on helping in the design Clinical Pathways.

Resum

Descriure els millors processos per a executar correctament una estratègia d'una forma eficient i amb qualitat no és sempre una Tasca fàcil. L'estandardització de processos en general i de Vies Clíniques en particular requerix de potents ferramentes de especificació i implementació que recolzen als experts en disseny. La utilització de models de Fluxos de Treball (de l'anglés *Workflows*) facilita als experts en disseny la creació les regles d'execució dels seus sistemes com si foren programadors. Encara així a causa de la gran mutabilitat dels processos reals és molt difícil conéixer com els processos s'estan executant en la realitat. La utilització de tècniques de reconeixement de formes poden ajudar a els experts en processos a inferir, a partir de mostres de Execució passades, models que expliquen la forma en què estos processos estan efectivament executant-se. Este paradigma és conegut com a Aprenentatge de Fluxos de Treball (de l'anglés *Workflow Mining*).

Els canvis d'estat en els processos d'atenció existent en les Vies Clíniques es basen en els resultats de les accions. Els Models actuals d'Aprenentatge de Fluxos de Treball no arrepleguen esta informació en els seus corpus. Per això, els actuals sistemes de aprenentatge no cobrixen les necessitats problemes complexos com és El cas de les Vies Clíniques.

En esta tesi es van a estudiar els models de representació, interpretació i aprenentatge de Fluxos de Treball amb la intenció De proposar un model adequat per a resoldre els problemes que Impedixen als dissenyadors de processos complexos, com a Vies Clíniques, Utilitzar tècniques d'Aprenentatge de Fluxos de Treball. Per a això se definirà un nou paradigma adequat per al suport al disseny de Vies Clíniques, a més de proporcionar les ferramentes adequades per al seu ús. Per açò en esta tesi es presenta a més un model de Representació de Fluxos de Treball amb una alta expressivitat i llegibilitat, una ferramenta de programari capaç d'executar i simular Fluxos de Treball per a crear corpus d'aprenentatge usant esta nova metodologia i un algoritme d'aprenentatge que siga capaç d'inferir Fluxos de Treball acords amb el problema del disseny de les Vies Clíniques.

Preámbulo

Actualmente es muy fácil encontrar empresas que inviertan cada vez más recursos en mejorar su gestión de la calidad. Para ello existen normativas que ayudan a avanzar en calidad a estas empresas mediante la mejora de sus procesos. Este es el caso de la conocida normativa *ISO 9001* que se basa en el estudio y la *estandarización* de los procesos para alcanzar una mayor calidad en la actividades generales de la empresa. Estandarizar un proceso es *elaborar, aplicar y mejorar* las normas que rigen el modo en el que se efectúan estas actividades.

La estandarización va a permitir que los procesos sean mejor entendidos por los responsables de las actividades de las empresas dándoles una valiosa información para la mejora de la calidad de las acciones realizadas. Esto permitirá mantener una relación de *buenas prácticas*, *predeterminar reacciones* ante problemas recurrentes o incluso facilitar la *trazabilidad* de los procesos, entre otras muchas ventajas.

Para conseguir una mayor efectividad en la estandarización, los procesos han de cumplir las siguientes condiciones: ser **formales**, es decir, que se puedan especificar en un lenguaje bien formado; **interpretables**, que puedan ser leídos por computadores; **expresivos**, que puedan representarse todas las situaciones según el campo de aplicación; **legibles**, que los expertos de la organización puedan entender la definición de los procesos y **trazables**, que la ejecución de los procesos pueda ser seguida por los actores de la organización.

Un complejo ejemplo de estandarización de procesos pueden ser las Vías Clínicas. Las Vías Clínicas son *protocolos de actuación estandarizados* en el campo de la medicina regidos por el *conocimiento médico* donde se detallan *secuencialmente* las actividades *clínicas*, de *soporte* y de *gestión* y coordinando a todos los *actores* implicados en el proceso: medicos, pacientes, trabajadores sociales, enfermeras, especialistas etc.

Las Vías Clínicas tienen como objetivos: mejorar las *capacidades funcionales* de los departamentos médicos, alcanzar una mayor *seguridad* en la practica clínica, mejorar la *coordinación* entre los profesionales de la salud que tratan al paciente, mejorar la *optimización* de los recursos y mejorar la *satisfacción de los pacientes*

Los precursores del Vias Clínicas surgieron como una herramienta de los profesionales de la enfermería para planificar los cuidados. Sin embargo, con la implicación posterior de los profesionales de la medicina, las Vías Clínicas se definen como una evolución de los casos Clínicos Médicos conforme a la estandarización de los procesos como si se tratasen de procesos industriales. En este modelo se prima la salud de los pacientes mediante la mejora del proceso asistencial de la manera más eficiente posible en el uso de los recursos y la planificación de las acciones en un tiempo establecido. Actualmente, más del 60 % de los hospitales de Estados Unidos trabaja con Vias Clínicas para los procedimientos más frecuentes.

Sin embargo, a pesar de la gran utilidad de las Vías Clínicas, estos protocolos se en-

cuentran con muchos problemas en la implantación. Las Vías Clínicas son muy complejas de definir y están ligadas al consenso de grupos de expertos en las materias correspondientes y cuentan con una alta carga subjetiva. Generalmente, los sistemas actuales donde se implantan Vías Clínicas se basan en el uso de formalismos basados en el rellenado de formularios estandarizados. Esta usual forma de implantar Vías Clínicas *burocratiza* el sistema hasta el punto de comprometer la correcta coordinación entre los profesionales.

La mayoría de las Vías Clínicas existentes están descritas en lenguaje natural. Esto complica la implantación de éstas debido, sobre todo, a la dificultad que entraña la automatización de modelos descritos de ésta forma. Por esto, es necesaria la utilización de modelos formales para describir Vías Clínicas que permitan la ejecución automatizada de estas.

Flujos de Trabajo

Las ventajas que la especificación formal de procesos puede aportar en el campo de la estandarización son multiples. Por un lado, la utilización de modelos formales facilita la creación de documentos con gran capacidad expresiva que permitan describir las especificaciones y restricciones de los problemas de una forma estandarizada. Por otro lado, este tipo de modelos posibilitan el uso de procedimientos automáticos de interpretación y aprendizaje de los modelos. De este modo, el primer punto a resolver en el campo de la estandarización de los procesos es la necesidad de encontrar un modelo formal capaz de representar y automatizar los procesos a definir. En este ámbito los Flujos de Trabajo (WF, del inglés *Workflows*) pueden resultar muy efectivos. Los WF son *especificaciones formales* de procesos que son diseñados para ser *automatizados*.

Un WF define acciones y transiciones para expresar el flujo que un proceso sigue. Estas acciones pueden ser realizadas por humanos que interpretan el WF descrito o automáticamente por sistemas informáticos.

Tradicionalmente, los WF eran utilizados para el modelado de procesos de negocio sencillos y repetitivos. Sin embargo, cada vez hay más campos complejos que investigan en la estandarización de procesos con WF, como son: los sistemas formales de guiado del cuidado del paciente (Vías Clínicas [51]), la automatización de cuestionarios de frecuencia alimenticia [111], la gestión automática distribuida de los procesos burocráticos de la administración pública [69] o la automatización de la recogida de datos de diferentes actores [98], entre otros.

La Tecnología de WF es un campo de investigación que está enfocado a la creación de lenguajes de especificación de automatización de procesos y de entornos de ejecución dinámica de estos.

Un ejemplo claro de esto es la utilización de sistemas de WF para la ejecución de Vías Clínicas. Tradicionalmente las Vías Clínicas están pensadas para ser utilizadas por actores humanos. Sin embargo, cada vez existen más sistemas de apoyo médico computerizado usando WF basados en este tipo de modelos.

Dos conceptos clave en el campo de la Tecnología de WF son la representación y la interpretación de WF. Por un lado, *representar un WF* es expresarlo de una manera formal para que este pueda ser automatizado. Representar un WF no implica que la implementación vaya a realizarse mediante sistemas software ya que esta puede ser ejecutada mediante actores humanos. Por otro lado, *interpretar un WF* es ejecutarlo automáticamente utili-

zando sistemas software. Para ello se necesita un Motor de WF que sea capaz de ejecutar procesos representados formalmente.

Para poder ejecutar procesos de negocio con suficientes garantías hay que contar con un buen modelo de representación e interpretación. Este modelo no solo debe permitir automatizar adecuadamente los procesos según el problema que se intente solucionar, sino también debe hacerlo de forma eficiente y legible.

Sin embargo, conseguir un compromiso entre la calidad de la representación y la eficiencia en la interpretación no es una tarea fácil. La calidad de la representación de WF se mide en términos diferentes a la calidad de interpretación de WF. En este contexto un buen modelo de representación de WF ha de ser **expresivo** y **legible**. Por otro lado un buen sistema de interpretación de WF debe ser lo más simple posible para ser leído por un sistema informático. Sin embargo, cuanta mayor expresividad tenga un modelo más complicado es limitar la complejidad para su ejecución. Además, para ser interpretado, un modelo ha de expresar información de bajo nivel que facilite la comunicación entre las piezas de software.

Por otra parte, al diseñar un WF, un experto plasma su conocimiento sobre el proceso en el modelo de representación. El diseño de WF es normalmente realizado de manera *manual* por expertos en procesos. Esto no es un problema cuando los procesos son sencillos de especificar, como puedan ser trámites burocráticos formalizados e inmutables, pero si lo es cuando los procesos no siguen un patrón conocido como es el caso del proceso que siguen las enfermedades sobre pacientes que tienen multiples patologías. Además el factor humano en el diseño, hace que este tipo de procesos diseñados tiendan a ser bastante *subjetivos*. Esto hace que los WF no se correspondan con la realidad debiendo ser continuamente readaptados.

En esta linea para un experto en procesos es crucial contar con buenas herramientas que le permitan conocer mejor los procesos en funcionamiento para así poder refinar los WF de forma iterativa.

Aprendizaje de Flujos de Trabajo

Una de las herramientas que pueden ayudar a los expertos en el diseño de los WF son las técnicas de reconocimiento de formas. El reconocimiento de formas [90] es una disciplina que consiste en el aprendizaje de modelos que clasifiquen la realidad utilizando herramientas estadísticas y/o sintácticas que infieran estos a partir de ejemplos de ejecución pasada. La utilización de técnicas basadas en la inferencia puede facilitar la tarea de estandarizar los procesos gracias a que pueden expresar los que se ejecutan en la realidad en un WF sin tener que diseñarlos desde cero y sin la subjetividad de los diseños realizados manualmente.

Los sistemas de Gestión de WF usados para interpretar, usualmente utilizan archivos donde se van almacenando las trazas de ejecución (WLogs, del ingles *Workflow Logs*) de las acciones realizadas por las instancias. Estos WLogs pueden emplearse para obtener corpus de entrenamiento que serán utilizados en las siguientes iteraciones del aprendizaje. En cada iteración del proceso se definirán procesos más precisos y menos sujetos a la percepción subjetiva de los expertos. Esta técnica se conoce en la literatura como Aprendizaje de Flujos de Trabajo (WM, del inglés *Workflow Mining*) [104]. Esta técnica facilita la estandarización en aquellos procesos complejos de diseñar. Las metodologías de

automatización de procesos pueden aprovecharse de estos sistemas para incrementar su eficacia.

En el caso de las Vías Clínicas, estas metodologías podrían ser usadas para aprender los flujos de ejecución de estas a partir de protocolos de cuidado ya ejecutados sobre pacientes anteriores. Sin embargo, la aplicación clásica estos sistemas tienen errores que no son admisibles en el caso de las Vías Clínicas debido a que comprometen la salud de los pacientes. Para solucionar problemas de este tipo, se han definido nuevas metodologías de aprendizaje basadas en la supervisión de expertos humanos como es el *Aprendizaje Adaptativo* o *Computer Assisted Learning* [23]. Intuitivamente, según esta metodología, cada vez que se aprende un modelo, un experto humano corrige los posibles errores ocurridos en la definición del WF antes de llevarlo a producción. En las Vías Clínicas, el médico siempre toma las últimas decisiones y nunca se dejan estas al albedrío de los sistemas de aprendizaje.

Las instancias de WF producidas por la ejecución de protocolos se pueden utilizar para entrenar cada vez mejores modelos de forma iterativa. Estos modelos pueden guiar a los facultativos en el cuidado de los pacientes de una forma estandarizada y con una menor de carga subjetiva. En estos casos, la ejecución del plan personalizado para un paciente específico significa crear una instancia de WF para ese paciente. Usualmente, cada instancia del mismo WF, se ejecuta de forma diferente al resto de pacientes tanto por las características personales de este (variaciones recogidas en el WF original) como por errores de la actual WF representado (variaciones no recogidas en el WF original). Esta instancia es continuamente supervisada por el médico. De hecho, el médico puede cambiar el orden y numero de acciones de ésta cuando sea necesario. Estos cambios anotados pasan a ser mejoras personalizadas para el paciente que no vienen reflejadas en el diseño original. Utilizando técnicas de WM con esta información podemos crear mejores modelos que cada vez sean más acordes con la realidad de la enfermedad para un mayor espectro de pacientes.

Estas anotaciones referidas a modificaciones de las instancias no siempre son fáciles de obtener. En el caso de las Vías Clínicas utilizadas por actores humanos, estas suelen ser anotadas en papel y no son accesibles. Además, únicamente se realizan arduas labores de recolección de estas excepciones a la vía cuando se realizan sesudos estudios clínicos médicos. Aún así, la dificultad de acceder a esta información hace que los casos clínicos suelan contar con un espacio muestral muy bajo. La automatización de los procesos mediante motores de WF va a permitir estandarizar los procesos de una forma más eficaz y sencilla, ya que el flujo de los procesos está siendo continuamente monitorizado y asistido por un sistema informático. Además de automatizar el proceso, estos sistemas permiten la recogida de las trazas de ejecución de los procesos para obtener corpus de entrenamiento que serán utilizados en las siguientes iteraciones del aprendizaje.

La mayoría de los algoritmos de WM que se hayan en la literatura están basados en el uso de WLogs transaccionales comerciales como muestras de entrada para la inferencia de WM. Esta aproximación se conoce como Minería de Flujos de Trabajo Basado en Eventos (EBWM, del inglés *Event-Based Workflow Mining*) [104]. Existen varios algoritmos citados en la literatura que están orientados al EBWM como el algoritmo α [104], el algoritmo *alpha+* [40], el Genetic Process Mining [41] o el Heuristic Miner [109]. El EBWM es una disciplina que utiliza el conjunto de eventos ocurridos durante toda la vida de cada una de las instancias de WF a modo de corpus para inferirlos. La información recogida en estos

corpus reduce a los eventos de inicio, y del fin, producidos por las acciones de los WF, sin tener en cuenta los resultados de dichas acciones. La razón por la que no utilizan esta información, es simplemente porque la mayoría de motores de WF no la ofrecen en sus WLogs. Esto es un problema en el caso de las Vías Clínicas. El flujo de los procesos que se ejecutan en las Vías Clínicas se basan en los resultados de las actividades en ejecución. Por ejemplo, en una Vía Clínica después de haber tomado la temperatura al paciente, las acciones siguientes dependen de los resultados de la acción. En este caso, dependiendo de si sus resultados son *temperatura normal* o *fiebre*, las siguientes acciones serían *no hacer nada* o *dar paracetamol*. La metodología del EBWM, no tiene en cuenta estos datos, y por lo tanto no es posible inferir WF que definan cambios en los procesos basados en el resultado de sus acciones. Por esto es necesario definir un nuevo método de WM que sea capaz de ofrecer una solución basada en inferencia a los modelos de estandarización de procesos cuyo flujo esté basado en los resultados de las acciones.

Hipótesis de Partida y objetivos de la Tesis

Dada la necesidad ya presentada de mejorar el modelo de WM actual para permitir a los sistemas de Vías Clínicas aprovecharse de las ventajas de este tipo de técnicas en su proceso de diseño, se va a proponer un nuevo paradigma de WM que recoja sus necesidades. En esta línea, partimos de la siguiente hipótesis:

> La utilización de métodos de inferencia inductiva aplicados a la estandarización de los procesos involucrados en una Vía Clínica a partir de corpus basados en los datos de inicio, fin y resultados de las acciones realizadas en los protocolos de cuidado de los pacientes permite el aprendizaje de modelos que describan el flujo de las actividades y las reglas de cambio que explican su funcionamiento

Según esta hipótesis, para poder conseguir algoritmos de inferencia basados en técnicas de WM capaces de aportar soluciones en el campo de las Vías Clínicas, es necesario mejorar los corpus actuales de WM de forma que incorporen los resultados de las acciones, además de la información de inicio y de fin. Esto nos lleva a formular el **objetivo principal** de la Tesis:

> Proponer un nuevo paradigma de Aprendizaje de Flujos de Trabajo basado en Actividades que utilice corpus que incorporen la información de inicio, fin y resultados de las acciones y aportar herramientas y algoritmos necesarios para implementar, evaluar y aprender sistemas basados en este paradigma.

Para poder probar el paradigma presentado y, a la postre, poder cumplir con el objetivo principal de la Tesis, se han marcado una serie de objetivos secundarios:

- Definir una herramienta de representación de Flujos de Trabajo acorde con el nuevo paradigma presentado, que permita representar fácilmente una gran cantidad de patrones de Flujos de Trabajo, que sea fácilmente legible, que sea fácilmente interpretable por sistemas de ejecución, y con una complejidad que facilite su inferencia utilizando técnicas de reconocimiento de patrones.
- Implementar un motor de interpretación capaz de ejecutar el modelo de representación presentado tanto desde un punto de vista de guiado de procesos, como desde un punto de vista de simulación de procesos. Este motor de interpretación deberá ser capaz de generar muestras con los datos necesarios para resolver el paradigma de Aprendizaje de Flujos de Trabajo basado en Actividades.
- Proponer un nuevo algoritmo de inferencia en el marco del paradigma de Aprendizaje de Flujos de Trabajo basado en Actividades.
- Proponer nuevas métricas para la validación de algoritmos de Aprendizaje de Flujos de Trabajo que permitan la comparación entre modelos basados en actividades y modelos basados en eventos.

Estos objetivos han servido de guía para la estructuración del trabajo durante el periodo de realización de la Tesis. Con esta Tesis se pretende la creación de un nuevo marco de apoyo al diseño de Vías Clínicas basado en técnicas de WM. Esto implica la creación de modelos de representación, interpretación y aprendizaje de WF acordes con el problema a solucionar, donde una alta legibilidad y gran expresividad deben convivir con una flexibilidad a la hora de la interpretación que permita la adaptación de los procesos a la variabilidad de los procesos de cuidado existente en las Vías Clínicas. Además, el sistema de interpretación creado no debe limitar la expresividad del modelo de representación y generar WLogs acordes con el modelo de WM basado en Actividades, almacenando la información adecuada. Por último, el algoritmo de inferencia creado ha de utilizar los WLogs creados por el motor de interpretación e inferir los flujos de las actividades, utilizando para ello el modelo de representación definido.

Estructura de la Tesis

La Tesis presentada se divide en tres partes bien diferenciadas, además de este preambulo. La distribución de los capítulos de la Tesis queda definida en la Figura 1.

Este preámbulo es el capítulo de introducción donde se expone el problema propuesto y se perfilan las soluciones propuestas identificando las hipotesis y objetivos de la Tesis.

La primera es la parte de antecedentes donde se exponen los estudios previos al trabajo realizado en la Tesis. Esta parte está a su vez dividida en cuatro capítulos: un capítulo

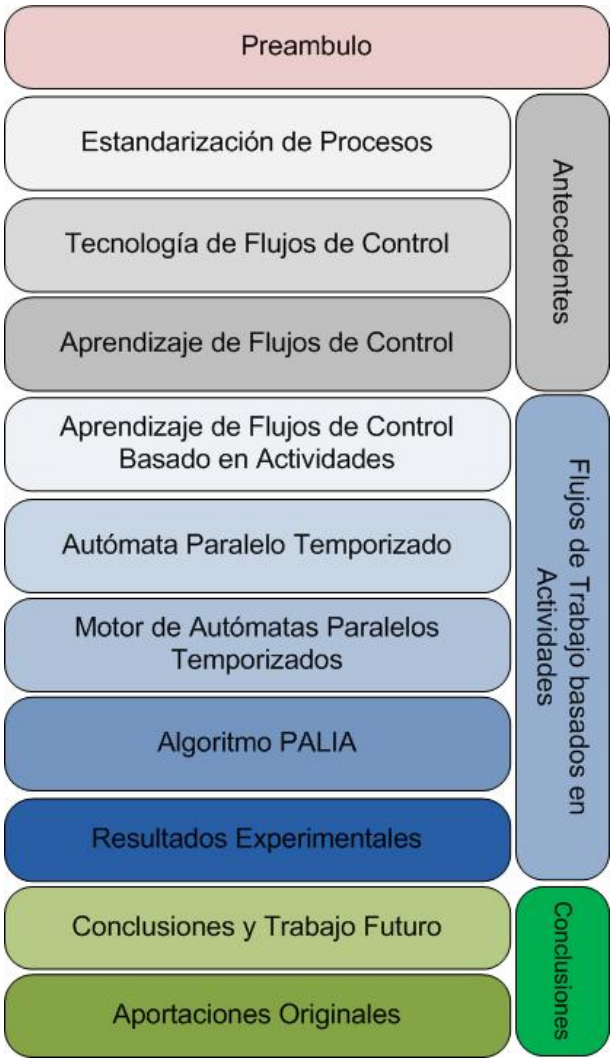

Figura 1: Estructura de la Tesis

dedicado a introducir el problema de la estandarización de procesos y el problema de las Vías Clínicas; otro que describe los modelos de formalización y normalización de procesos mediante WF; uno más que realiza un estado del arte en sistemas de WF; y, por ultimo, un capítulo dedicado a los estudios previos en aprendizaje inductivo de WF a partir de muestras.

En la segunda parte se exponen los principales resultados obtenidos en la Tesis distribuidos en cuatro capítulos. En el primero de los capítulos se expone la propuesta del nuevo paradigma de Aprendizaje de Flujos de Trabajo Basado en Actividades, que resulta una evolución del paradigma basado en eventos y que cubre las necesidades de un gran numero de campos de la estandarización de procesos como es el caso de las Vías Clínicas. El siguiente capítulo presenta un nuevo modelo para la representación de Flujos de Trabajo formalmente definido denominado TPA. Este modelo resulta muy expresivo y cuenta con complejidad gramatical regular. El tercer capítulo de esta parte presenta un motor de de Flujos de Trabajo basado en TPA llamado TPAEngine, capaz de generar WLogs basados en actividades capaz de funcionar con pocos recursos y con un sistema auxiliar destinado a simular la ejecución de los flujos para validar los WF antes de su implantación y crear corpus de entrenamiento. El cuarto capítulo de esta parte presenta un algoritmo de Aprendizaje de Flujos de Trabajo basado en Actividades llamado PALIA que aprende WF representados en forma de TPA a partir de WLogs generados por el TPAEngine. El quinto capítulo de esta parte está orientado a presentar los resultados experimentales de la Tesis; donde se realizan una serie de experimentos con corpus de basados en WLogs tanto existentes en la literatura como generados. Estos WLogs se utilizan para comparar los algoritmos existentes en la literatura con el algoritmo PALIA utilizando métricas adecuadas.

La tercera y ultima parte de la Tesis presenta las conclusiones finales evaluando las hipótesis y objetivos cumplidos, y describe las posibles líneas de trabajo futuro. Además, esta parte incluye un capítulo donde se exponen las principales aportaciones originales, publicaciones y proyectos asociados.

Parte I

Antecedentes

Capítulo 1

Estandarización de Procesos

Dentro de este capítulo se expondrán los conceptos básicos a tener en cuenta en la estandarización de procesos y se identificarán los problemas clave que se encuentran en esta disciplina. Por último, se propondrá un ejemplo explícito de problema de estandarización de procesos basado en el problema de las Vías Clínicas.

1.1. Introducción a la Estandarización de Procesos

En la sociedad industrializada actual cada vez es más común la búsqueda de técnicas que permitan realizar los procesos de negocio de la manera más eficiente posible. Según Davenport [39] un proceso de negocio (del inglés *Business Process*) es:

> *Un conjunto de actividades estructuradas y medidas diseñadas para producir una salida específica para un cliente particular o mercado.*

Más adelante, la Workflow Management Coalition, en su glosario [110], define proceso de negocio como:

> *Un conjunto de procedimientos o actividades interconectadas que en conjunto realizan un objetivo de negocio o una política estratégica normalmente en el contexto de una estructura organizacional donde están definidos roles funcionales y relaciones.*

Mas informalmente, los procesos de negocio son el conjunto de las *actividades necesarias* para realizar las *tareas específicas* que nos van a permitir crear los productos finales dentro de una empresa u organización. Por un lado, se especifican los procesos operativos, que son los que efectivamente generan los productos, como pueden ser la venta, la fabricación o el marketing. Por otro lado, también son especificados los procesos de dirección, como pueda ser la estrategia o los procesos de soporte como la contabilidad.

Estos procesos han de ser lo más eficiente posibles manteniendo su eficacia de modo que para obtener los mismos resultados en cuanto a producción y calidad se consuman los mínimos recursos posibles. Por ello, se definieron teorías que permitieran a las empresas

mejorar sus procesos mediante la Gestión de la Calidad Total o TQM (Total Quality Management) [49]. Este modelo presigue la mejora continua de la calidad desde un punto de vista holístico, buscando un mejor conocimiento y control de todo el sistema. Esto usualmente se aborda estudiando y mejorando los procesos de negocio internos hasta conseguir productos de calidad de una forma eficiente.

En esta linea, para mejorar los procesos es necesario formular las reglas por las que estos procesos se van a regir. Lo que llamanos *estandarizar* los procesos. De este modo, para que los procesos de negocio sean aplicados de una forma óptima han de ser **estandarizados** [38]. Según la Organización Internacional de la Estandarización:

> *La estandarización es el proceso de formular y aplicar reglas con el propósito de realizar en orden una actividad específica para el beneficio y con la obtención de una economía de conjunto óptimo teniendo en cuenta las características funcionales y los requisitos de seguridad. Se basa en los resultados consolidados de la ciencia, la técnica y la experiencia. Determina no solamente la base para el presente si no también para el desarrollo futuro y debe mantener su paso acorde con el progreso.*

Informalmente, *estandarizar un proceso* es crear un protocolo o conjunto de normas que rijan la ejecución de los procesos de un modo recomendado u obligatorio. La estandarización de estos procesos va a permitir no solo entender mejor los procesos, sino que:

- Mantiene una relación de *buenas prácticas* que describen los procesos más eficientes para realizar las acciones, tanto ordinarias como extraordinarias, de la organización.
- Permite predeterminar las reacciones ante cualquier acción previa producida.
- Permite la trazabilidad de los procesos detectando errores en las primeras etapas de los procesos.
- Permite delimitar fronteras para la realización de los procesos de una forma especializada, permitiendo a seleccionar no solo a los mejores expertos internos para las distintas acciones sino que permite la externalización de los procesos para que sean realizados por terceras organizaciones.

La estandarización de procesos también requerirá cambios en la estrategia. Una vez los procesos estandarizados sean usados día a día y sean interiorizados por los actores involucrados, los resultados de estos serán predecibles y podrán ser mejor controlados desde la administración. De los ejecutivos dependerá la revisión de los planes estratégicos y competitivos para priorizar los procesos distintivos sobre los demás en función de los intereses de la empresa.

Un ejemplo de proceso estandarizado puede ser el proceso de ventas de una empresa. En la Figura 1.1 se puede ver un gráfico que ilustra el ejemplo. En este caso el cliente formaliza una petición de pedido. El departamento de ventas recoge el pedido y pide a contabilidad un informe de morosidad del cliente, si es favorable emite una orden de envío al departamento de logística, en caso contrario, rechazará el pedido. La estandarización de

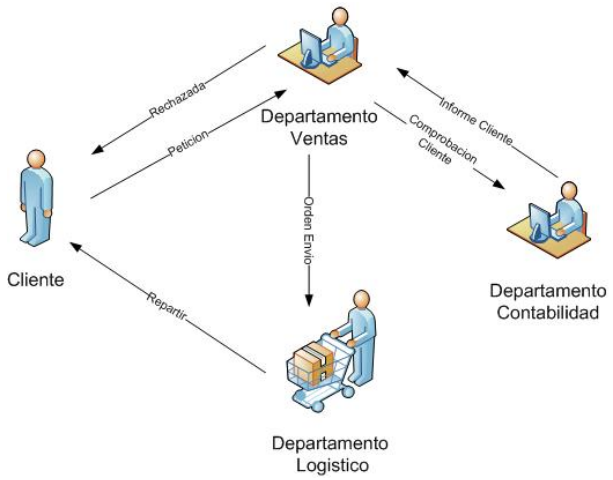

Figura 1.1: Ejemplo de estandarización de procesos

procesos como estos, permite la normalización de plazos de respuesta y de entrega, lo que repercute en el servicio al cliente. Además en caso de problemas es mucho más fácil detectar en que puntos e están produciendo retrasos y solucionarlos mediante la modificación del estándar.

1.2. Implantación de procesos de negocio

La estandarización de procesos no es un trabajo fácil de realizar. Los procesos realizados en una organización no son siempre fácilmente *visibles* y *reconocibles*. Para recoger este conocimiento hay que contar con herramientas que permitan expresar de una manera legible y expresiva los distintos procedimientos.

Además, un estándar de procesos solo tiene impacto si todo el mundo lo adopta, por lo que se deben hacer grandes esfuerzos no solo por obtener los estándares sino también por implantarlos. Por un lado, las herramientas de representación de han de ser multidisciplinares para permitir que los distintos actores englobados en los procedimientos de una organización puedan entender y aplicar los procesos estandarizados de la forma menos traumática posible. Por otro lado, hay que monitorizar los protocolos para que saber el estado en el que se encuentran los procesos en cada momento para permitir la detección de problemas de implantación [37].

En esta linea, el éxito de la estandarización de procesos en una organización va a depender en gran parte de la capacidad de especificar procesos que:

- Sean *formales*, es decir, que permitan su especificación de una manera lo más formal posible. Esto va a permitir reducir la ambigüedad y la subjetividad de los documentos de estandarización de procesos.

- Permitan la *automatización de los procesos*. La estandarización de los procesos tiene que permitir que los procesos especificados sean repetibles por los actores de una organización

- Sean *expresivos*, es decir, que los modelos puedan expresar todas las situaciones que se puedan dar en la especificación de los procesos de negocio de la organización
- Sean *legibles*, es decir, que todos los actores de la organización, entiendan los procesos descritos y comprendan sus implicaciones.
- Permitan *trazabilidad*, es decir, que puedan ser monitorizados para observar su evolución y detectar errores en la implantación de los procesos.

El campo de la estandarización de procesos es muy amplio. En esta memoria se va a abordar el problema de la estandarización en un campo muy concreto y especialmente sensible a los problemas de implantación de los procesos estandarizados como son las Vías Clínicas.

1.3. Vías Clínicas como procesos estandarizados

La medicina es uno de los campos donde la estandarización de procesos está cada vez más en auge. Tradicionalmente, la medicina ha sido un campo en el que sus profesionales han ido creando sus procesos individualmente y unilateralmente, de forma independiente del resto de la comunidad médica. De este modo, es muy fácil encontrar que ante los mismos problemas, existan profesionales que aborden las soluciones desde puntos de vista totalmente diferentes [14]. Esta variabilidad en la atención médica a menudo oscurece el efecto de una intervención en particular en el cuidado médico en cuestión. Por esto, la estandarización de procesos clínicos se está adquiriendo cada vez más auge en entre los sistemas de salud actuales.

1.3.1. Medicina Basada en la Evidencia

A principio de los 90 nació un nuevo enfoque para la práctica de la medicina que intentaba resolver estos problemas aplicando el método científico a la investigación y enseñanza en la medicina. Este nuevo paradigma se denominó Medicina Basada en la Evidencia [97].

Sacket et al [89] define la Medicina Basada en la Evidencia como:

> *El uso racional, explícito, juicioso y actualizado de la mejor evidencia científica aplicado al cuidado y manejo de pacientes individuales. La práctica de Medicina Basada en la Evidencia requiere la integración de la experiencia clínica individual con la mejor evidencia externa derivada de los estudios de investigación sistemática*

Más adelante Elstein [47] dice de la Medicina Basada en la Evidencia que:

> *Practicar la Medicina Basada en la Evidencia significa integrar la competencia clínica individual con la mejor evidencia clínica externa disponible a partir de la investigación sistemática*

La Medicina Basada en la Evidencia pretende que los procesos de gestión clínica de los pacientes utilicen como método para tomar decisiones clínicas la *búsqueda* en la literatura biomédica original y la *lectura crítica* de esta basada en la *experiencia clínica* del profesional. De este modo, las características principales de la Medicina Basada en la Evidencia son:

- *El uso intensivo de la literatura biomédica.*- La creación y continua mejora de la literatura médica va a permitir crear mejores repositorios donde encontrar evidencias estadísticas útiles sobre determinadas enfermedades y sus tratamientos. En esta literatura no solo se encontrarán tratamientos médicos, sino también ensayos clínicos exitosos estadísticamente demostrados que pueden aportar ideas para el establecimiento de protocolos de cuidado.
- *La lectura crítica de la literatura basada en la experiencia clínica individual.*- Debido a la variabilidad de los pacientes en medicina, es posible encontrar casos muy diferentes ante enfermedades similares. Sin embargo, la decisión final sobre la selección del cuidado del paciente la ha de tomar el profesional de la medicina basándose en su experiencia clínica.

Tradicionalmente, la medicina se ha basada a menudo en casos clínicos que no contaban con una evidencia estadística aceptable. La Medicina Basada en la Evidencia pretende ser complementaria a la aproximación tradicional aportando un marco de evidencias fiable que permita a los facultativos una aplicación de la medicina de una forma más estandarizada.

1.3.2. Guías Clínicas

Una de las herramientas más usadas por parte del paradigma de la Medicina Basada en la Evidencia son las Guías Clínicas. Una Guía Clínica es un documento cuyo objetivo es apoyar al guiado de las decisiones y los criterios propios de la practica médica en áreas específicas del cuidado de la salud apoyando el diagnostico, gestión y tratamiento de la enfermedad.

Una Guía Clínica se puede definir como [51]:

> *Declaraciones desarrolladas sistemáticamente para asistir a los profesionales de la salud y a las decisiones de los pacientes sobre el cuidado apropiado de la salud en circunstancias clínicas específicas.*

Las Guías Clínicas modernas son complejos documentos que identifican, resumen y evalúan el conocimiento basado en la evidencia médica en cuanto a los datos de prevención, diagnosis, prognosis y terapias incluyendo dosis de medicamentos con su riesgo/beneficio y coste/efectividad asociado. Estos complejos documentos constituyen un método de **estandarización de procesos clínicos**. Estos documentos son la referencia que los profesionales de la medicina suelen utilizar para informarse de los cuidados estandarizados a los pacientes en una determinada enfermedad.

Estas normas resultan a la postre recomendaciones que apoyan a los profesionales de la salud o incluso a los pacientes a mejorar la gestión de la enfermedad permitiéndole

seleccionar las mejores opciones diagnósticas y terapéuticas para una condición clínica específica. Las Guías Clínicas en sí mismas son un complejo documento que es creado por el consenso de los órganos de gobierno de los sistemas de salud en los diferentes países, por lo que suelen tener ámbito nacional o regional. En la actualidad existen muchos organismos que se encargan de crear y mantener Guías Clínicas para la comunidad médica como por ejemplo la *National Guideline Clearinghouse de Estados Unidos*[1] o la *National Institute for Health and Clinical Excellence*[2] de Reino Unido o la *German Agency for Quality in Medicine*[3] de Alemania.

1.3.3. Vías Clínicas

A partir de la Guías Clínicas, a veces incluso formando parte de ellas, se definen las Vías Clínicas. De un modo formal, una Vía Clínica se puede definir como [50]:

> *Una herramienta que, sin querer reemplazar el imprescindible criterio clínico de los profesionales, pretenden contribuir a hacer un uso más adecuado, racional y coordinado de los recursos sanitarios existentes y a mejorar la calidad de la atención prestada*

Otra definición de Vía Clínica más orientada a la gestión de los recursos es [93]:

> *La Vía Clínica ayuda a realizar una utilización máxima y eficiente de los recursos humanos y materiales con el objetivo de proporcionar unos cuidados médicos de alta calidad minimizando los costes*

En general una Vía Clínica es una herramienta de gestión multidisciplinar basada en planes asistenciales para guiar un proceso de curso clínico predecible en la que se detallan secuencialmente las actividades clínicas, de soporte y de gestión además de las responsabilidades de los profesionales implicados.

La diferencia principal entre Vías Clínicas y Guías Clínicas es su ámbito de aplicación. Mientras que las Guías Clínicas están orientadas a definir recomendaciones clínicas generales de cuidado, tratamiento y diagnóstico, las Vías Clínicas están específicamente pensadas para coordinar a todos los actores de los procesos de cuidado y de gestión de un modo explícito. A menudo las Vías clínicas se forman a partir de las Guías Clínicas pasando a ser parte de estas.

Actualmente existen varios organismos y bibliotecas digitales que se encargan de crear y mantener Vías Clínicas. Varias de éstas se pueden ver en repositorios de documentación Fisterra [52], la biblioteca Pubmed [76] o la Cochrane [25]

En la Figura 1.2 se define una Via Clínica simple para el cuidado de la Diabetes. En ella se define la forma en la que los fármacos han de ser administrados en este caso según el paciente vaya evolucionando.

[1] http://www.guideline.gov/
[2] http://www.nice.org.uk/
[3] http://www.aezq.de/aezq/english/switchLanguage?set_language=en

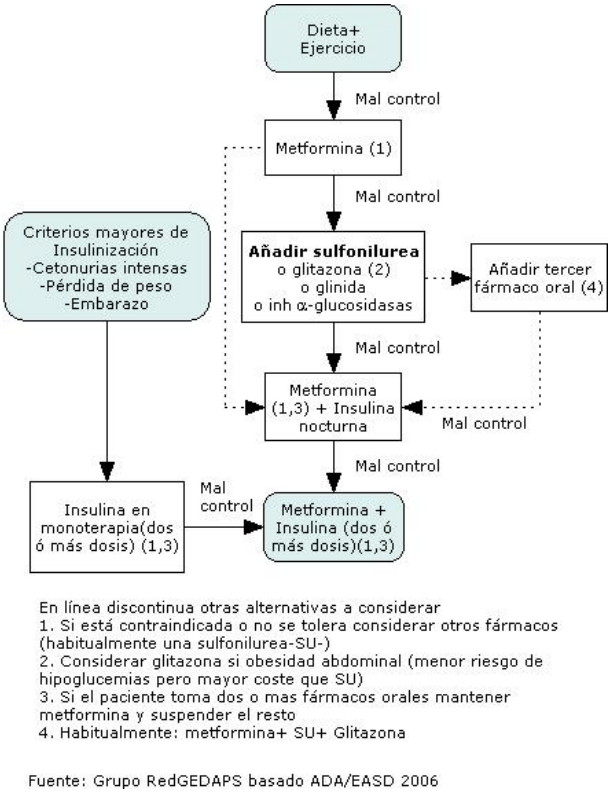

Figura 1.2: Ejemplo de Vía Clínica del cuidado de la diabetes de tipo 2

Ventajas de las Vías Clínicas

Las Vías Clínicas resultan una potente herramienta para la estandarización de procesos clínicos. Esta estandarización ofrece un conjunto de ventajas a la medicina:

- **Facilitar la praxis de los profesionales de la salud**. No solo para los expertos en el campo en cuestión, sino para que tanto profesionales de la salud de especialidades totalmente heterogéneas, como médicos de familia o estudiantes de medicina puedan contar con guías de actuación que les permitan tomar decisiones con una mayor eficacia y seguridad cuando se encuentren, con pacientes que cursen con estas enfermedades protocolizadas.
- Forzar a los expertos a **unificar criterios** que eliminarán la nociva variabilidad de la práctica clínica.
- **Apoyar la gestión administrativa del cuidado** además de la propia gestión clínica. Esto va a permitir preveer los costes administrativos que va a suponer el tratamiento de un determinado paciente, lo que a influir muy positivamente en la mejora de la gestión de las unidades de salud.

Problemas en la estandarización de Vías Clínicas

Se ha demostrado en los puntos anteriores la utilizad de las Vías Clínicas y Guías Clínicas. Sin embargo existen varios problemas que complican la creación de estos documentos:

- Las Vías Clínicas se definen creando grandes documentos, planillas con protocolos de actuación, recomendaciones, guías de tratamiento, etc. Estos grandes documentos son muy difíciles de realizar porque han de ser creados a partir de un duro proceso iterativo que es realizado por consenso entre un grupo de expertos en un determinado campo que está influenciado por la subjetividad de sus miembros. Además, en en caso de las Vías Clínicas también se incorporan los procesos burocráticos y de gestión que hacen más complicada su definición.
- El momento de la implantación de las Vías Clínicas es muy delicado. En este momento es cuando se han de adaptar las Vías Clínicas a cada hospital, ya que los procesos tanto clínicos como burocráticos son propios de cada sistema de salud. Las Vías Clínicas suelen ser personalizadas hasta tal punto que cada centro tenga la suya propia.
- La ejecución de las Vías Clínicas complica más el proceso. Dado que aún son muchos los sistemas de salud que aún funcionan en papel, la coordinación entre los servicios utilizando Vías Clínicas hacen que los procesos tengan tanta carga burocrática que resultan imposibles de ejecutar.

Debido a estos problemas, el diseño e implantación de Vías Clínicas se hace muy complicado. Para facilitar este proceso sería necesario proporcionar herramientas que permitan recoger información estadística sobre los procesos realizados sobre los pacientes y utilizar esta información para realizar estadísticas que permitan crear nuevo conocimiento que podría entonces ser incorporado a la vía.

En este trabajo se va a proponer un método de apoyo al diseño y estandarización de las Vías Clínicas basado en inferencia[4]. Para poder conseguir este propósito, el primer paso sería encontrar un modelo que nos permitiese implantar las Vías Clínicas donde se registren las acciones realizadas mediante sistemas informáticos. Esto permitiría recoger más fácilmente los datos que servirían para descubrir los patrones de la ejecución de procesos. Además, estos patrones de ejecución deberían permitir a los facultativos diseñar fácilmente Vías Clínicas para su posterior utilización en la estandarización de los cuidados mediante sistemas de guiado y supervisión automática.

[4]Se entiende como inferencia a el proceso de aprendizaje inductivo automático a partir de muestras de ejecución pasadas.

Capítulo 2

Tecnología de Flujos de Trabajo

En este capítulo, primero se va a describir una tecnología para estandarizar y automatizar procesos denominada Flujos de Trabajo. Seguidamente se va a exponer el problema de su representación e interpretación. Finalmente se analizaran modelos de Flujos de Trabajo existentes centrándose en sus ventajas e inconvenientes para su aplicación al diseño de Vías Clínicas.

2.1. Definición de Flujos de Trabajo

El diseño de procesos por parte de expertos es un problema de gran complejidad. Por otro lado, si además de representar el proceso se quiere interpretar, el problema se agrava aún mas. Según el modelo tradicional, si se quieren aprovechar los sistemas informáticos para supervisar y apoyar la implantación de los procesos diseñados es necesario crear un software que lo haga específicamente. Por ello, cuando un experto en procesos, como un médico, quiere poner en marcha un proceso, ha de recurrir a un programador para que le construya el programa adecuado. Además, un experto en procesos pueden decidir en cualquier momento cambiar el flujo de ejecución para hacerlo más adecuado, lo que requeriría de nuevo la mediación del personal informático. Esta mediación supone a la postre errores y retrasos en la implantación de procesos muy simples.

Este problema es resoluble dotando a los diseñadores de procesos de herramientas para poder diseñar sus propios procesos. Para ello, es necesario encontrar modelos que nos permitan realizar de un modo natural la automatización de los procesos para poder estandarizarlos. En el campo de la informática existe una disciplina que se encarga de la investigación en lenguajes de especificación de automatización de procesos y de entornos de ejecución dinámica de estos. Esta disciplina se denomina Tecnología de Flujos de Trabajo. La tecnología de Flujos de Trabajo provee de lenguajes para definir procesos de una forma estándar que, de ser suficientemente formales y recoger la suficiente información, permitirían incluso la ejecución automática de los procesos en sistemas informáticos. El principal objeto de esta disciplina son los llamados Flujos de Trabajo.

Un Flujo de Trabajo (WF, del inglés *Workflow*) es la especificación formal de un proceso diseñado para ser automatizado. Un WF define acciones y reacciones entre estas para estandarizar la planificación de un proceso. Estas acciones pueden ser realizadas por humanos que interpretan el WF descrito o realizadas automáticamente por sistemas informáticos. Más formalmente, la *Workflow Management Coalition* define WF como [110]:

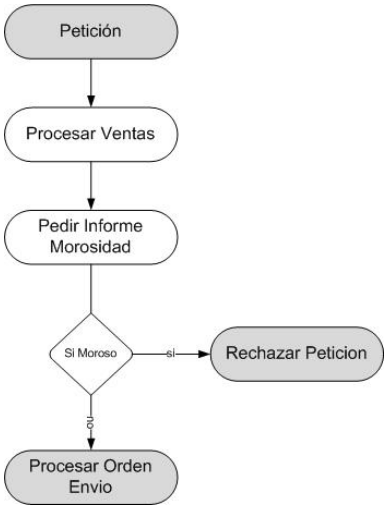

Figura 2.1: Ejemplo de Flujo de Trabajo que describe el proceso estandarizado de ventas

> *La automatización de un proceso de negocio, en su totalidad o en parte, cuando documentos, información o tareas se pasan de un actor a otro, para realizar una acción, de acuerdo con una serie de reglas de procedimiento.*

Tradicionalmente, los WF eran utilizados para el modelado de procesos de negocio sencillos y repetitivos. Estos WF eran sencillos de representar, como por ejemplo los procesos burocráticos simples. Sin embargo, con la llegada de la cultura de la estandarización, nuevos procesos más complejos en están empezando buscar ayuda en la tecnología de WF, como es el caso de las Vías Clínicas.

En la Figura 2.1 se muestra un WF que define el proceso ejemplo de ventas del capítulo anterior. En el se definen secuencialmente las acciones que tienen lugar en el proceso, como la petición del informe de morosidad, y las decisiones a tomar dependiendo de los resultados de las acciones, como el rechazo de la petición o el procesamiento de la orden de envío dependiendo de la morosidad del cliente.

Los WF están pensados para resolver problemas de estandarización de procesos mediante la definición de estos de una forma no ambigua y pensada para su replicación. Sin embargo, no siempre es buena la utilización de WF. Por ejemplo, un programa informático que nunca vaya a modificarse es preferible implementarlo con un lenguaje de programación estándar que utilizando WF, ya que, los WF resultan menos potentes que dichos lenguajes, además de ser mucho menos eficientes. De este modo, la utilización de sistemas de WF viene indicada para el diseño de procesos repetitivos que:

- Necesiten una legibilidad a un alto nivel, por ejemplo, porque han de ser diseñados por expertos sin conocimientos de programación. Por ejemplo, en el caso de una Vía Clínica, son los propios profesionales de la medicina los que deben diseñar los procesos.

- Necesiten un control del flujo especifico que permitan en cualquier momento localizar el estado actual del flujo y los siguientes caminos a seguir. Por ejemplo, en una Vía Clínica puede ser necesario saber el estado actual del paciente, y que pasos podría seguir en el flujo normal.

- Contemplen la posibilidad de modificar una instancia del proceso para que se ejecute de distinta manera a como había sido planeada en un principio. Por ejemplo, la modificación de una Vía Clínica en tiempo de ejecución puede ser necesaria por complicaciones de una enfermedad. Esta característica potencia la variabilidad haciendo que existan instancias de WF que se ejecuten de una forma totalmente diferente a como fueron programadas.

- Ofrezcan un guiado de la tarea en contraposición con una actuación autónoma, es decir, que necesiten confirmación de los pasos a seguir. En el caso de las Vías Clínicas, estas suelen ser vistas como una guía de actuación médica en la que se definen pasos que pueden ejecutarse de un modo autónomo, o que están dirigidas para guiar la actuación del profesional sanitario o de gestión, por lo que necesitan aportar constantemente información sobre el estado y el camino seguido por el paciente en la instancia del WF.

En este trabajo, se van a utilizar modelos basados en WF para definir Vías Clínicas. La utilización de modelos de WF para la creación de Vías Clínicas va a permitir que los propios medicos diseñen los procesos inherentes al cuidado de las enfermedades. Sin embargo, los modelos de WF a utilizar han de tener una capacidad expresiva muy alta, ya que los procesos de la vida real pueden llegar a ser muy complejos. Por otro lado, los WF deben evitar la ambigüedad, no solo para que actores humanos puedan ejecutar los WF, sino también para que puedan ser automatizados en sistemas informáticos. En ente caso, seleccionar un buen modelo de diseño y ejecución de WF resulta vital para que la implantación de las Vías Clínicas tenga éxito.

2.2. Representación de Flujos de Trabajo

Uno de los problemas más importantes a la hora de hacer posible el diseño y ejecución del WF, es la representación del WF. Los lenguajes de especificación de WF deben ser suficientemente expresivos y lo suficientemente fáciles de representar para permitir a expertos en sus respectivas áreas modelar los procesos de la realidad de una manera formal. De hecho, la *representación de WF es una descripción formal de un conjunto de procesos y sus reglas de cambio para que este pueda ser automatizado ya sea por sistemas informáticos como por actores humanos.* Representar un WF significa definir un modelo de forma no ambigua con la suficiente información para ser repetido siguiendo siempre el mismo esquema.

Existen innumerables modelos y lenguajes de representación de WF que pueden ser utilizados como herramientas para diseñar procesos de negocio como BPMN [59], Diagramas de Actividad UML [13] o XPDL [24] que serán analizados más adelante. Los sistemas de gestión de WF comerciales suelen venir acompañados de completas utilidades gráficas para ayudar al máximo al experto a diseñar sus procesos. Estas herramientas gráficas suelen potenciar la facilidad de la descripción de los WF ya que facilitan la legibilidad y

hacen más entendible el sistema. El mayor problema de las herramientas comerciales es que suelen pecar de estar demasiado pensadas para solucionar los problemas de ejecución de WF más que los problemas de representación por lo que pecan de falta de expresividad. Por otro lado, aunque también hay modelos más orientados a resolver el problema de la representación, que pueden venir en forma de estándares o iniciativas individuales que intentan aportar solución a problemas generales o específicos de distintos entornos, estos suelen tener problemas a la hora de la ejecución, ya que estos lenguajes suelen ser más difíciles de interpretar debido a su complejidad.

2.2.1. Características de los lenguajes de representación de Flujos de Trabajo

Para representar un WF, hay que definir un lenguaje capaz de expresar todas las situaciones que requiera el problema a resolver, de la forma más entendible posible. El amplio ámbito de aplicación de los lenguajes de representación de WF hace que unos modelos puedan ser validos en unos entornos e inválidos en otros. Por ejemplo, entornos que necesiten una gran legibilidad, pueden restringir la expresividad para conseguir un lenguaje valido.

La evaluación de los lenguajes de representación de WF ha sido hasta hace relativamente poco tiempo un problema difícil de resolver. Esto ha sido debido, no solo a la dificultad del problema sino tambien a su subjetividad. En este punto se van describir las características más importantes de los lenguajes de Representación de WF, junto con sus métricas evaluadoras más usadas. Esto nos servirá de punto de partida para la elección de modelos de representación de WF dentro de los distintos entornos de aplicación

Expresividad

La expresividad es *la capacidad que tiene un lenguaje para representar diferentes patrones en un WF*. De este modo, cuanto más patrones sea capaz de plasmar un lenguaje mejor será su expresividad. Ejemplos de patrones a representar usando WF son, *acciones paralelas*, es decir, poder ejecutar dos acciones concurrentemente, o *sincronizaciones de acciones*, es decir, ramas paralelas que se fusionan en una solo bajo unas condiciones de sincronización.

Existe una iniciativa [77] para la creación y el mantenimiento de distintos patrones de comportamiento que podrían ser incorporados en la definición de WF. Esta iniciativa, parte de la base publicada por Will van der Aalst en [105] donde se publicaron los 20 patrones básicos que todo Sistema de Gestión de Workflows debería optar a cumplir. Estos patrones se han convertido en una medida de la expresividad de los modelos de representación de WF, de modo que cuantos más patrones sea capaz de expresar un lenguaje, más expresivo será. Esta iniciativa, no solo crea, mantiene y revisa estos patrones, sino que actúa de observatorio sobre las herramientas de gestión de WF activas en el mercado, realizando una evaluación continua de estas.

Estos patrones, llamados Patrones de WF, no solo están orientados a la representación de WF, sino también al modo en que estos se ejecutan y la información y los recursos entre las distintas acciones es compartida. Existen cuatro categorías de Patrones de WF:

- *Patrones de Control de Flujo*.- Estos patrones se corresponden con una revisión de

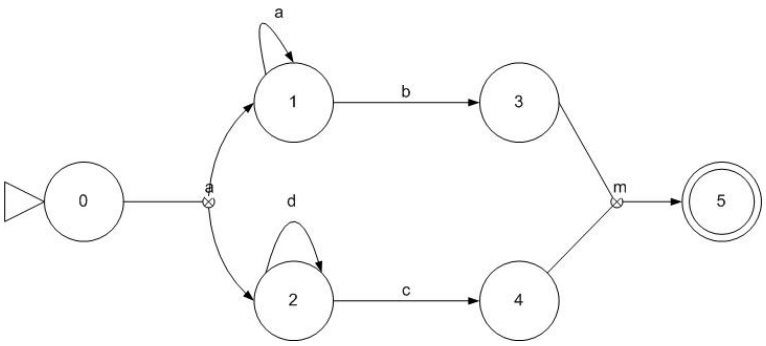

Figura 2.2: Ejemplo de Workflow con patrones de control de flujo

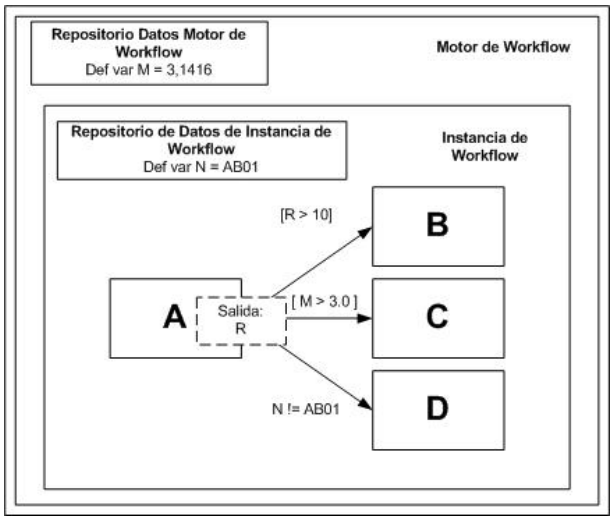

Figura 2.3: Ejemplo de Workflow con patrones de datos

los patrones de control de flujo presentados por Will van der Aalst en [87]. Estos patrones definen situaciones de gestión del flujo de procesos de negocio, tales como sincronizaciones o separación paralela de procesos. En la Figura 2.2 se puede ver un ejemplo de WF en el que se pueden identificar varios patrones de control de flujo, entre ellos un patrón de *separación paralela*, donde los procesos *1* y *2* se ejecutan paralelamente después de ejecutarse el proceso *0*; y, análogamente, un patrón de *sincronización paralela* donde después de ejecutar los procesos *3* y *4* se ejecuta el proceso *5*.

- *Patrones de Datos*.- Presentados en [85], estos patrones ofrecen una serie de conceptos que pueden aplicarse a la utilización de datos en los sistemas de WF. Estos patrones no solo definen la forma en la que los datos pueden emplearse dentro de los WF y los datos que los sistemas de gestión de WF puede usar, sino que también caracterizan la interacción de los elementos de datos con otros WF o sistemas externos. En la Figura 2.3 se presenta un ejemplo de patrón de WF de datos, en el se presenta una rutina basada en datos, por la cual, la transición al siguiente estado depende de operaciones sobre los datos; ya estén estos localizados en el repositorio de datos del motor de WF, en el de la instancia de WF, o incluso a la salida del proceso ejecutado. En la Figura, se puede observar como el paso de la acción A a las acciones B, C y D, está supeditado a variables dependientes de la instancia en ejecución (N) o incluso a variables globales a todas las ejecuciones de WF como M.

- *Patrones de Recursos*.- Estos patrones [86] están centrados en el modelado de recursos y su interacción con sistemas de gestión de procesos de negocio. Estos recursos pueden ser humanos, o no humanos, como equipamientos, salas, etc. En el ejemplo del proceso de ventas, cuando se producía un pedido, desde el departamento de ventas se pedía un informe de morosidad. En este caso, el responsable del departamento de contabilidad podría ser tratado desde el WF como un recurso compartido, que requeriría un control que gestione el acceso a este. Este tipo de control de los recursos, son los estudiados por el conjunto de los patrones de recursos.

- *Patrones de Manejo de Excepciones*.- Estos patrones definen soluciones al manejo de las excepciones ocurridas durante el la ejecución de WF. Estos patrones fueron presentados en [88]. En el ejemplo del proceso de ventas si el formulario de petición del informe de morosidad está incompleto o es erróneo, se producirá una excepción en el proceso. El protocolo de acciones a realizar en caso de que esto se produzca, es el ámbito de aplicación los patrones de manejo de excepciones.

Los Patrones de Control de Flujo, son los patrones más usados para medir la expresividad de los modelos, esto es debido, a que son los que definen la coordinación de las acciones, mientras que los demás describen situaciones de bajo nivel más dependientes de la implementación de los procesos que del flujo en si.

Este trabajo, está orientado a la utilización de WF para la definición de Vías Clínicas. El problema de diseño de Vías Clínicas que se va a abordar en este documento está orientado únicamente al flujo que las acciones siguen. No se van a abordar por tanto ni el ámbito de las variables de paso, ni el control de los recursos, ni el manejo de las excepciones que no están explicitamente representadas en el proceso en si. Por ello, en este trabajo se van a utilizar solo los Patrones de Control de Flujo para medir la expresividad. En el

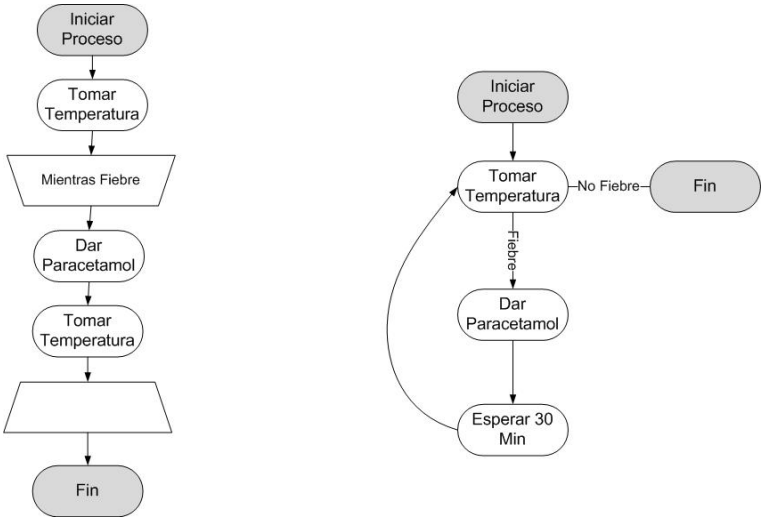

Figura 2.4: Representación de un Workflow con estructuras de programación y con Autómata Finito

Anexo A se describen con más detalle los Patrones de Control de Flujo que serán utilizados para la evaluación de los modelos.

Legibilidad

A pesar de la importancia de la expresividad en el diseño de procesos, no es suficiente para que la implantación de un sistema de WF tenga éxito. Los WF están pensados para ser usados por personal que no tenga conocimientos en programación, es decir para que sean los propios expertos en procesos los que generen su propias descripciones.

La legibilidad es *la facilidad con la que un experto puede entender el flujo definido en una especificación de WF*. Los modelos de WF actuales suelen contar con representaciones gráficas que facilitan la legibilidad del problema.

La legibilidad de un WF es un valor muy subjetivo. Modelos que suelen ser legibles para programadores informáticos no suelen serlo para expertos externos. Por ejemplo, en la Figura 2.4 se muestra un proceso diseñado de dos formas diferentes. La primera se realiza utilizando estructuras de flujo complejo como la estructura de programación *mientras*, por otro lado la segunda, solo utiliza estados y flechas. Entre estos dos WF un programador informático consideraría más legible la primera, mientras que un médico puede sentirse más cómodo con modelos más parecidos al de la segunda.

Dejando a un lado estas diferencias subjetivas, otro factor que afecta sobre la la legibilidad es la talla del modelo. Si un modelo utiliza más estructuras que otro para representar el mismo proceso, esto hace que su legibilidad se vea reducida. Por ejemplo, en la Figura 2.5 se ve el ejemplo anterior representado con una Red de Petri. La Red de Petri (que analizaremos con más detalle más adelante) es mucho más expresiva que los autómatas finitos, sin embargo, su legibilidad es menor porque requiere más estructuras para definir los procesos.

En el problema que nos ocupa vamos a tomar en cuenta que los diseñadores de WF

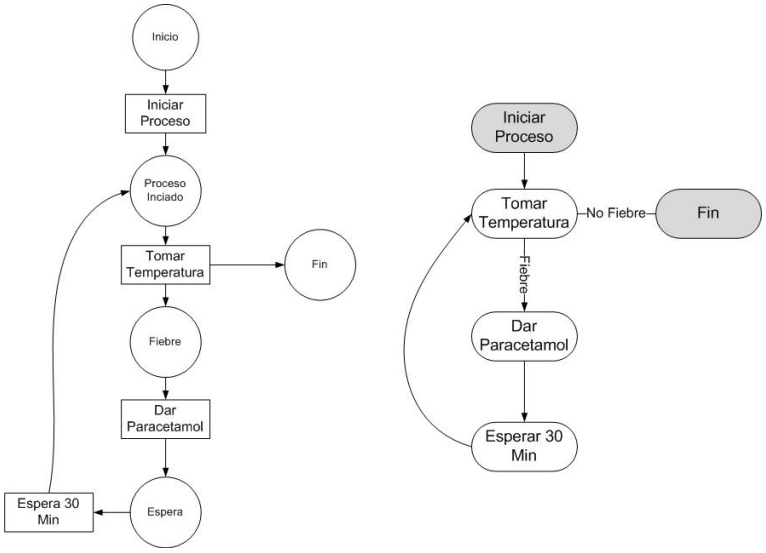

Figura 2.5: Representación de un Workflow con Redes de Petri y con Autómata Finito

serán los propios médicos y se va a dar mayor prioridad a los modelos que incorporen modelos más sencillos para este tipo de expertos, y que la talla de los flujos diseñados sea menor.

Complejidad gramatical

Mientras que la legibilidad se refiere la complejidad del lenguaje con respecto al experto humano que la trata, la complejidad gramatical se refiere a la complejidad inherente al modelo de representación en sí mismo. La complejidad gramatical de un WF está determinada por sus capacidades de transición, de su dispositivo de memoria y las capacidades de inspección en su memoria. Es bien conocido en la teoría de la compilación que la complejidad gramatical del lenguaje de representación marca la facilidad con la que el código descrito es ejecutado [10]. La complejidad va a marcar facilidad con la que el lenguaje de representación de WF va a ser interpretado. A mayor complejidad del lenguaje de representación será más complicada su ejecución. Para el problema que nos ocupa se va a primar la utilización de lenguajes sencillos y fáciles de ejecutar por encima de los demás, para evitar al máximo los problemas de la implantación de los modelos.

Modelado del tiempo

Además de ser capaz de representar flujos de trabajo, los procesos a diseñar pueden requerir patrones basados en el tiempo. La representación del tiempo es un concepto muy útil en la mayoría de los entornos de uso de WF. Los expertos pueden utilizar WF para representar procesos en los que se requiera realizar esperas, tras las cuales se realizan acciones especificas, crear acciones periódicas, o incluso empezar tareas a horas prefijadas. Algunos ejemplos del uso del modelado del tiempo son:

- Los modelos de gestión de tráfico requieren que el sistema espere durante un periodo de tiempo, normalmente unos segundos, antes de pasar al siguiente ciclo.
- En los modelos de Vías Clínicas, es necesario esperar a que un medicamento surta efecto antes de tomar ninguna iniciativa. Esto haría que el flujo pudiera detenerse hasta varios dias antes de seguir con el siguiente paso.
- En un sistema de Inteligencia Ambiental, es posible que se necesite un reloj que despierte a la persona todos los día a una determinada hora.

La utilización del tiempo es un concepto muy ampliamente usado, y muy necesario en la definición de los modelos de WF. Por ello es extremadamente importante que los lenguajes de definición de WF permitan de un modo u otro expresar el flujo del tiempo en sus modelos.

2.2.2. Vías Clínicas y Representación de Flujos de Trabajo

En general, la utilización de modelos de WF en el ámbito de las Vías Clínicas puede proporcionar muchas ayudas para que sean los propios facultativos quienes describan sus procesos. Sin embargo, para que esto tenga éxito, el lenguaje de representación ha de ser capaz de representar todas aquellos patrones requeridos por los protocolos de cuidado médico, así como manejar en tiempo de una forma sencilla. Esto ha de realizarlo de una forma legible, que permita al facultativo no solo aprender fácilmente el diseño sino que las estructuras sean espacialmente compactas para limitar la talla de los procesos diseñados. Por otro lado, el lenguaje debe ser gramaticalmente lo más simple posible para facilitar su compilación y ejecución.

2.3. Interpretación de Flujos de Trabajo

Una vez definido un lenguaje de Representación de WF adecuado el siguiente paso es su interpretación. Interpretar un WF es *ejecutar las acciones del WF según las reglas definidas en la representación asociada utilizando para ello sistemas informáticos*. La ejecución de WF es un problema de compromiso entre la complejidad del modelo de representación y las necesidades específicas de los usuarios que quieren ejecutar los WF. La interpretación del los WF requiere que los lenguajes de representación sean formales, lo suficientemente sencillos para ser ejecutados y carezcan de ambigüedad. Muchos lenguajes orientados a la representación resultan intuitivos y legibles pero que limitan el control sobre la ejecución del proceso ya que son menos formales y más ambiguos. Por otro lado, podemos encontrarnos con lenguajes más orientados a la interpretación que permiten un control total de la ejecución del proceso mediante lenguajes formales, pero que resultan menos legibles y por lo tanto son más difíciles de utilizar.

Los WF se diferencian de los sistemas de programación imperativa usuales en que los motores de interpretación de WF posibilitan *saber en que punto del proceso se encuentra el WF* en cada momento. Además, sería posible *cambiar el flujo de ejecución de una instancia del WF dinámicamente* según las necesidades del problema.

La ejecución de WF se basa en la generación de *instancias de WF* para cada ejecución individual. Cada una de estas instancias coordinan la ejecución de los procesos para

cada caso particular siguiendo uno de los caminos posibles definidos en el WF según de la instanciación de la reglas en cada momento. Estas reglas se encuentran inicialmente definidas en una *plantilla de WF*. Esta plantilla representa el diseño inicial del WF. En el se definen las variables, las acciones, las transiciones y las reglas de cambio de estado. A partir de esta plantilla se crean las instancias. Las plantillas y las instancias son análogas a lo que en programación orientada a objetos se define como clase y objeto. Las plantillas serían como las clases que define todo el proceso, y las instancias serían como los objetos que controlan el flujo de una ejecución.

Un sistema de interpretación de WF, o motor de WF es un componente software que toma como entrada un WF y mantiene el estado del ejecución de los procesos delegando y distribuyendo las actividades a realizar de los procesos entre actores humanos y aplicaciones software. Dicho de otro modo, un sistema de interpretación de WF es un *Software* capaz de *tomar como entrada un WF diseñado en un lenguaje de representación formal* y *ejecutar los procesos incluidos* conforme a las reglas definidas en dicho lenguaje.

2.3.1. Representación e Interpretación

Existe una diferencia clara entre los lenguajes diseñados para representar WF para ser automatizados por humanos que los específicamente diseñados para ser interpretados. Como demuestra la práctica en implementación de sistemas de interpretación de WF, los lenguajes que son muy legibles por expertos humanos, resultan a menudo complejos y por tanto difíciles de ejecutar. Por otro lado, los sistemas de gestión muy legibles tienden a ser poco expresivos, y los muy expresivos, tienden a ser poco legibles y manejables solo por personal altamente entrenado. Esta dificultad a la hora de escoger un lenguaje que incorpore este trío de características hace que se definan arquitecturas pensadas para coordinar lenguajes legibles, usualmente basados en modelos gráficos, para definir los WF que se traducen en lenguajes más sencillos que son fácilmente interpretables.

Usualmente los modelos de interpretación de WF se presentan definiendo una arquitectura para diseño y ejecución de procesos separado en capas. En la Figura 2.6 se presenta un esquema representativo del funcionamiento general de un sistema de Interpretación de WF. Cada uno de los elementos de la arquitectura se presenta a continuación:

- *Capa de interfaz gráfica.*- Para facilitar la legibilidad de la definición de procesos, los sistemas de diseño de WF cuentan con sistemas gráficos basados en primitivas para la especificación de WF. Esto, facilita a los usuarios no versados en el conocimiento de la programación el diseño procesos basados en estos sistemas.

- *Capa de Programación Lógica.*- Como ya se expuso anteriormente, la representación gráfica de WF tiene la ventaja que permite una definición de WF sencilla al alcance de cualquier experto en procesos de negocio que no tenga nociones de programación. Sin embargo, estas definiciones, distan mucho de las necesarias para la creación de sistemas formales para la especificación de WF. La capa de Programación Lógica está pensada para tratar con este problema. En esta capa se definen parámetros de configuración y las primitivas que serán usadas por la capa de interfaz gráfica para definir los WF.

- *Capa de Representación de WF.*- Esta capa se encarga de juntar la información gráfica de WF, junto con la información parametrizada de este, para especificar

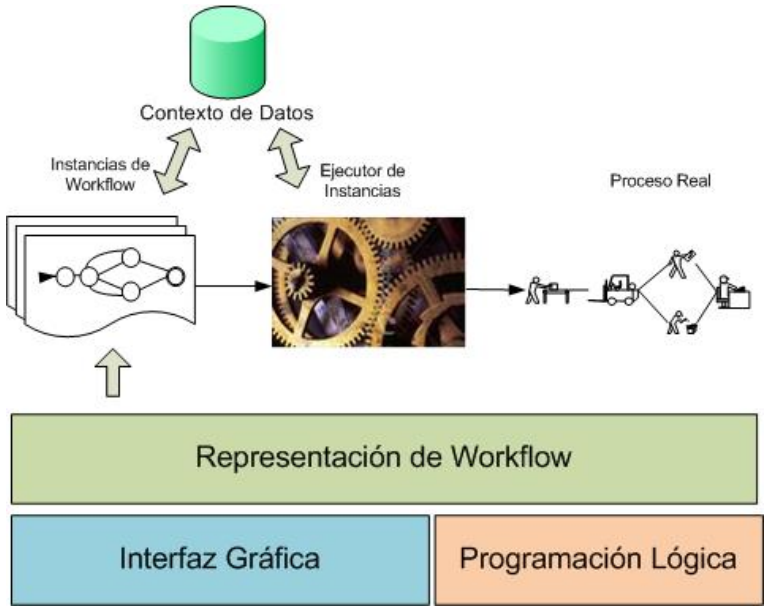

Figura 2.6: Arquitectura general de interpretación de Workflows

formalmente WF que puedan ser ejecutados. Los Diseñadores de procesos dibujan los WF utilizando las librerías gráficas existentes en la capa de interfaz gráfica, que pueden estar asociadas a librerías de ejecución de acciones sitas en la capa de Programación lógica. Por ejemplo, una de las acciones de la capa de interfaz gráfica podría ser enviar un SMS, la capa de programación lógica contendría las librerías necesarias para enviar el mensaje. Por otro lado, la capa de interfaz gráfica cuenta con iconos gráficas que representan el envío de mensajes SMS. Estos iconos son incorporados a la descripción del WF diseñando el proceso de una forma visual.

- *Ejecutor de Instancias de WF.*- Esta capa es el núcleo de ejecución de los procesos. Cada una de los WF especificados en la capa de representación son traducidos a un lenguaje sencillo preparado para ser interpretado que y porteriormente ejecutados por el ejecutor de instancias tras ser instanciados. Este elemento es el finalmente coordina la ejecución ordenada de las acciones según esta marcada por las reglas definidas en el WF.

- *Capa de Contexto de Datos.*- Esta capa almacena los datos que las instancias de WF generan en tiempo de ejecución. Esta capa es útil para permitir la compartición de datos entre instancias así como la gestión de las variables de entrada y salida que requieran los WF.

Este modelo general de representación e interpretación de WF puede sufrir cambios debido al campo de aplicación de los WF. Por ejemplo, en entornos donde el diseño de WF no va a ser realizado por expertos externos sino que es realizado por profesionales informáticos, la capa de interfaz gráfica puede ser eliminada. En este caso los programadores

definen los WF utilizando directamente lenguajes formales orientados a la interpretación de procesos.

2.3.2. Problemas de la interpretación de Flujos de Trabajo

La interpretación de WF no es un problema fácil de solucionar. Es tal la cantidad de dominios en los que la automatización de los procesos de negocio tiene aplicación, que es difícil para los desarrolladores de estos sistemas definir un modelo general que sea capaz de cumplir todas las características requeridas por este tipo de aplicaciones. Algunos de los desafíos más importantes que un diseñador de motores de interpretación de WF tiene que abordar son listados a continuación.

Diseño expresivo, sencillo y legible

El compromiso entre la expresividad y la sencillez en el campo de la ejecución de WF es un problema que requiere un estudio del ámbito de aplicación del WF para encontrar la mejor solución. Existen muchos modelos de lenguajes de representación de WF con una expresividad aceptable para ser usados. Sin embargo, muchos de estos lenguajes, normalmente basados en representación gráfica, suelen ofrecer una pobre información sobre la ejecución de los procesos que es necesaria a la hora de la automatización del WF. Muchas veces la información necesaria para ejecutar es de tan bajo nivel, que es comparable a la programación imperativa, y gráficamente puede ser tediosa de programar. Esto hace que las empresas de software encargadas de diseñar sistemas de gestión de WF hayan limitado la expresividad gráfica de muchos de sus lenguajes, para hacerlo más legible en un primer nivel, e incorporar código fuente directamente en algunas partes del WF que permitan suplir las carencias de expresividad. Este modelo permite un diseño de alto nivel proporcionado por expertos que es complementado por programación imperativa proporcionada por los desarrolladores software.

Sin embargo, esto no es deseable en todos los casos por lo que hay que buscar sistemas que sean legibles y puedan aportar información para la ejecución de una forma sencilla sin perder por ello expresividad.

Comunicación con procesos externos

Muchas de las actividades a realizar dentro de un WF pueden necesitar acceso a procesos externos nativos o distribuidos por la red para poder llevar a cabo sus funciones. Acceso a bases de datos, comunicación con aplicaciones nativas o acceso a Web Services son algunos de los ejemplos de lo que un sistema de gestión de WF puede necesitar para ejecutar las actividades que tenga programadas. Este problema suele ser solucionado por los sistemas software permitiendo la incorporación de código nativo en la especificación del WF. No obstante, dentro de este marco, se han desarrollado arquitecturas software orientadas a la solución de este tipo de problemas promoviendo la utilización de herramientas software de comunicación que permiten la intercomunicación de aplicaciones heterogéneas, este tipo de utilidades se denominan Motores de Integración o Enterprise Service Buses (ESB) [22]. Los principales motores de interpretación del mercado cuentan con implementaciones de estos buses para facilitar la interconexión de los sistemas legados con las nuevas aplicaciones.

Monitorización de instancias de Flujos de Trabajo

Para un sistema de interpretación de WF no es suficiente con hacer que las instancias se ejecuten, sino que muchas veces es necesario monitorizarlas. Usualmente en los sistemas de WF es necesario poder acceder a la instancia para saber en que estado se encuentra, y en caso de problemas permitir el desbloqueo o la cancelación de la instancia. Además, este tipo de herramientas van a permitir a los diseñadores de WF a detectar errores muy difíciles de depurar en tiempo de diseño. Los sistemas de interpretación de WF suelen proveer de programas de monitorización que permiten seguir la ejecución de las instancias

Modificación de Instancias de Flujos de Trabajo

Muchas de las aplicaciones de WF requieren que se pueda modificar la instancia de WF en tiempo de ejecución. Esto es útil en casos complejos donde no todas las ejecuciones de las instancias de WF se expresan según el WF original debido a que no representa fielmente el modelo real. Esto que puede parecer una característica fácil de implementar, tiene muchos problemas a nivel práctico. En primer lugar, la modificación de una instancia de WF en tiempo de ejecución puede traer consigo muchos problemas colaterales, como salto de excepciones por incongruencias ente los procesos y la necesidad de implementación de patrones de ejecución complejos, como cancelación de actividades.

Modificar la instancia obliga a cancelar el proceso de automatización y personalizar *manualmente* el proceso. Esto puede significar diseñar completamente un WF solo para una instancia para hacerla más adecuadas a la realidad. Estas modificaciones individuales pueden ser anotadas y ser utilizadas para mejorar el WF original. De esta forma, casos similares serían tenidos en cuenta por este en el futuro. Esto forma un ciclo de especificación en espiral que involucra al usuario en la creación de los modelos, donde en cada ciclo se introduce una mejora sobre la especificación del WF actual.

Las empresas diseñadoras de sistemas de WF suelen transformar las especificaciones gráficas en código para una ejecución más eficiente. Es por esto que modificar este código en tiempo de ejecución es una ardua tarea que muchas empresas han optado por no permitir, limitando mucho el potencial de los sistemas de WF.

2.3.3. Vías Clínicas e Interpretación de Flujos de Trabajo

Para permitir la automatización y la recogida de datos de Vias Clínicas basadas en WF es necesario disponer de un buen motor de ejecución de WF capaz de ejecutar el modelo de representación seleccionado. Este motor ha de permitir el guiado y la supervisión de la implantación de los procesos de cuidado descritos. Esto implica que debe existir un marco de ejecución adecuado para el lenguaje de representación que admita la automatización de patrones de WF especialmente complejos de la manera más eficiente posible.

Por otro lado, en las Vías Clínicas, los procesos de cuidado en ejecución pueden cambiar en cualquier momento debido a la variabilidad de los pacientes. Esto implica que las reglas de ejecución de los procesos puedan verse modificadas en cualquier momento, lo que exige que el sistema sea capaz de modificar las instancias en tiempo de ejecución

Capítulo 3

Sistemas de gestión de Flujos de Trabajo

Tras haber marcado las necesidades en cuanto a representación e interpretación de WF el siguiente paso es la selección de un sistema adecuado para el problema en cuestión. En este capítulo, se van a analizar distintos sistemas de WF usados para representar e interpretar WF. De un modo general, para interpretar WF se suelen utilizar lenguajes próximos a los lenguajes de programación porque resultan fáciles de ejecutar. Por otro lado, para representar WF se utilizan lenguajes mucho más gráficos, muchas veces basados en metáforas, que permiten un diseño intuitivo y expresivo. Esta importante diferencia entre lenguajes orientados a la representación y lenguajes orientados a la interpretación hace necesaria la utilización de traductores para utilizar las características de ambos lenguajes.

Los modelos a estudiar se pueden separar en tres grupos diferentes: a) los modelos formales, que son modelos teóricos utilizados para la formalización de procesos; b) modelos basados en la representación, que tienen como base la creación de modelos donde se prima la expresividad y legibilidad; y c) los modelos basados en la interpretación, cuyo fin es la creación de modelos de WF fáciles de ejecutar en sistemas informáticos.

3.1. Modelos formales

En esta sección se evaluarán una serie de modelos formalizados teóricamente descritos para la descripción y formalización de procesos. Estos modelos, aunque pueden no contar con herramientas software asociadas, pueden ayudar al diseño de nuevos lenguajes y herramientas aportándoles un marco formal que permita su evaluación.

3.1.1. Redes de Petri

Las Redes de Petri, presentadas por Karl Adam Petri, son una aproximación matemática para modelar sistemas distribuidos de un modo natural. La principal contribución de las Redes de Petri a la representación de procesos es su gran expresividad. Las Redes de Petri tienen suficiente expresividad para representar todos los patrones de WF [106]. Además, las Redes de Petri pueden modelar sincronización de procesos, eventos asíncronos, operaciones concurrentes y compartición de recursos. Es por ello que las Redes de Petri

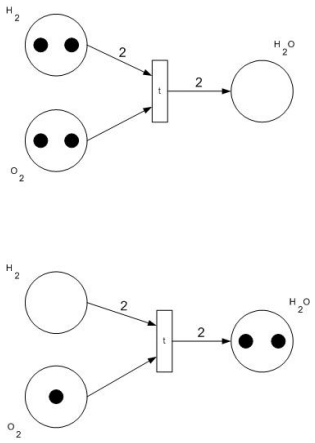

Figura 3.1: Ejemplo de Red de Petri modelando el proceso de formación del agua

son el marco formal más ampliamente usado para formalización y simulación de procesos muy complejos.

La principal aportación de las Redes de Petri con respecto a los modelos de estados finitos comunes, es que permite la estructuración de procesos que requieren coordinación en la ejecución de actividades, expresando fácilmente modelos que requieren sincronía, además de englobar todo esto dentro de un marco de análisis formal que permite obtener información del comportamiento dinámico del sistema modelado.

Definición 3.1 *Una Red de Petri[72] es una 5-tupla PN={P, T, F, W, M_0} donde:*
$P = \{p_1, p_2, ..., p_m\}$ *es un conjunto finito de estados,*
$T = \{t_1, t_2, ..., t_n\}$ *es un conjunto finito de transiciones,*
$F \subseteq (P \times T) \cup (T \times P)$ *es un conjunto de arcos,*
$W : F \rightarrow \{1, 2, 3, ...\}$ *es una función de pesos,*
$M_0 : P \rightarrow \{0, 1, 2, 3, ...\}$ *es el marcado inicial,*
$P \cap T = \emptyset$ *y* $P \cup T \neq \emptyset$.

Esta definición puede explicarse gráficamente con un ejemplo. En la Figura 3.1, ejemplo recogido de [72], se puede observar una Red de Petri que define el proceso de formación del agua a partir de hidrógeno y oxígeno. Esta Red de Petri está formada por tres estados (places) H_2, O_2 y H_2O. Según esta figura, se forman dos moléculas de agua (H_2O) a partir de dos moléculas de Hidrógeno (H_2) y una de oxígeno (O_2). Los puntos negros en los estados, se conocen como marcas. En la primera red del ejemplo se produce primero lo que se define como un marcado {2,2,0} respectivamente a los estados basado en el número de átomos existentes de cada uno de los elementos. La cardinalidad de las transiciones viene dado por lo que se conoce como pesos y definen el número de marcas necesarias para para realizar una transición. La red de abajo en la Figura, representa en marcado final al ejecutar la transición {0,1,2} de donde se deduce que al final quedan una molécula de oxígeno y dos de agua.

Redes de Petri y representación de WF

Las Redes de Petri son extremadamente útiles para la especificación y simulación de procesos complejos que requieran paralelismos y concurrencias de procesos. Existen diferentes extensiones de Redes de Petri que aportan muchos tipos de funcionalidad para llevar a cabo complejos patrones de WF. Estas extensiones son comúnmente denominadas Redes de Petri de alto nivel. Algunas de estas redes son:

- *Redes de Petri Etiquetadas*, presentadas en [65], son aquellas donde cada una de las transiciones viene etiquetada con una acción. Usando este modelo, se podrían diseñar WF donde cada marcado supusiera un estado, y cuando un determinado marcado cumpliera unas condiciones determinadas se ejecutaría la transición al mismo tiempo que se pondría en marcha la acción designada por su etiqueta

- *Redes de Petri Coloreadas*, presentadas en [66], son compatibles hacia atrás con el modelo original. La diferencia fundamental que existe entre las redes y las originales es que en las Redes de Petri Coloreadas es que en estas las transiciones vienen etiquetadas con un valor asociado denominado token. Estos tokens son tipados utilizando programación clásica y utilizados como parámetros de entrada y salida de las transiciones. Este modelo, se podría automatizar del mismo modo que el modelo etiquetado, con la ventaja de proveer en cada transición no solo de la acción a realizar sino también de los parámetros necesarios para ejecutarla.

- *Redes de Petri Jerárquicas*, presentadas en [80], son una extensión de las Redes de Petri que permite la modelización de transiciones como subtareas. En una Red de Petri Jerárquica, existe un flujo principal en el que cada transición puede representar una Red de Petri, que es ejecutada cuando se ejecuta la transición. Este modelo de *sub-red*, permite la reutilización de Redes de Petri comunes en un programa que pueden ser representadas en varias capas, lo que mejora la legibilidad de la Red de Petri.

- Las *Redes de Petri Temporales*, presentadas en [83], son Red de Petri que asignan a cada una de las transiciones de la red un numero real no negativo que define el tiempo de disparo de la transición. Este tiempo define el intervalo de tiempo que ha de pasar entre que la transición se ha habilitado, y la transición se efectúa. Este sistema permite incorporar el modelado del tiempo en las Redes de Petri.

- Las *Redes de Petri Estocásticas*, presentadas en [63], están pensadas para su utilización en modelado y simulación de procesos. Estas redes pueden verse como una extensión de las Redes de Petri Temporales a las que se le añade un factor de aleatoriedad para su ejecución en simulación. De este modo, se puede simular un comportamiento pseudo-real en una Red para tener una mayor información sobre el funcionamiento del sistema completo.

En el campo de las Vías Clínicas, un ejemplo de Red de Petri que formaliza la ejecución de una Vía se propone en 3.2. En este ejemplo se presenta un flujo en el que existen dos actores, un médico y un paciente. El paciente entra en el sistema y realiza un test, cuando el médico entra, revisa el test, si el resultado de este es *OK* realiza la *acción 1* y en caso contrario realiza la *acción 2*. Seguidamente el proceso termina. Los círculos de esta

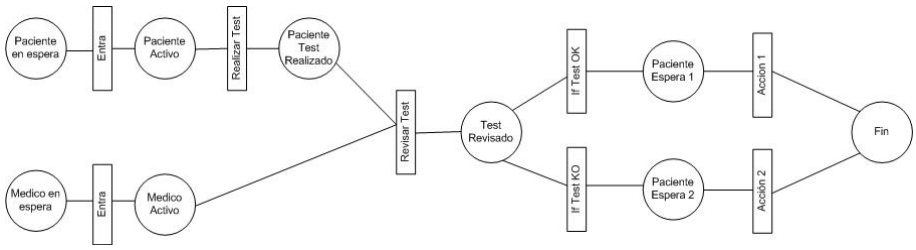

Figura 3.2: Ejemplo de Via Clínica modelada en Red de Petri

Figura son los como estados de espera mientras que las figuras rectangulares representan los eventos de transición entre los estados.

Existen gran cantidad de sistemas software para el diseño de Redes de Petri tales como XPetri [56] que proporciona un editor gráfico, herramientas de simulación y de análisis estructural de Redes de Petri, EZPetri [9] que proporciona el mismo tipo de herramientas para el entorno de programación Eclipse, o PetriSim [92] que es capaz de simular además Redes de Petri temporales. Desde el punto de vista de los WF existen modelos de interpretación basados en Redes de Petri como el YAWL.

YAWL (del inglés *Yet Another Workflow Language*), presentado en [103], es un lenguaje definido y diseñado específicamente para cumplir los patrones de WF de Van der Aalst. De hecho, está diseñado a partir de estos, por el mismo grupo de investigación que diseñó los patrones de WF. Además del lenguaje de representación, se definió un modelo de ejecución basado en una extensión de las Redes de Petri y una herramienta gráfica de definición de WF.

Uno de los problemas de implementación de YAWL son algunas limitaciones que ha sido necesario realizar en el modelo de interpretación para la implementación de WF. Según Van der Aalst et all [103], ha sido necesario limitar la implementación del lenguaje en algunos aspectos, ya que la ejecución de Redes de Petri de alto nivel es complicada debido a que tienen una complejidad gramatical muy alta. Además, a pesar de ser un modelo altamente expresivo, la mayor desventaja de este modelo es que su definición gráfica tienen menos legibilidad y su ejecución es bastante ineficiente.

Ventajas y limitaciones de las Redes de Petri

Las Redes de Petri son un herramienta ampliamente usada para simulación y modelado de sistemas concurrentes o que requieren sincronía. Sus múltiples extensiones permiten definir modelos extraordinariamente expresivos para multiples casos. De hecho, las Redes de Petri Coloreadas son capaces de expresar de una forma natural todos los patrones de WF presentados. Además de esto, es capaz de modelar el tiempo fácilmente, gracias a la extensión dedicada a ello.

Sin embargo, dada la gran complejidad de las Redes de Petri (Turing Completa) su ejecución no es un tema trivial. Según Hofstede y Van der Aalst [103], las Redes de Petri son más complejas en la interpretación que ninguno de los sistemas Software de gestión de WF actuales. Sin embargo justifican su utilización por estar basado en estados, su semántica formal, y su abundancia de técnicas analíticas [61]. Esta complejidad se basa

en el concepto de marcado multiple. El marcado múltiple ofrece información sobre la ejecución de multiples hilos dentro de un único proceso. Este concepto es poco intuitivo por lo que tiende a no ser utilizado, utilizando en su lugar modelos basados en Redes de Petri Seguras. Las Redes de Petri Seguras son Redes de Petri, cuyo marcado esta limitado a como máximo uno en cada estado. Como se verá más adelante las Redes de Petri seguras son, con mucho, menos complejas que las Redes de Petri completas [65]. Es por esto que en la práctica las Redes de Petri se convierten en manos de los expertos en un modelo de representación de WF demasiado complejo para la expresividad requerida. Además, la mayoría de las extensiones de las Redes de Petri, que, no obstante, resultan vitales para crear procesos de WF útiles, requieren de conocimientos de algorítmica para la su definición, cosa que puede resultar totalmente restrictiva para su utilización por parte de dichos diseñadores.

3.1.2. Autómata Finito Paralelo

Para definir WF, los diseñadores de procesos tradicionalmente suelen utilizar diagramas de flujo de datos como herramienta gráfica de especificación. Los diagramas de flujo de datos son modelos lógico-gráficos para representar procesos y flujos de negocio que fueron diseñados para facilitar la tarea de especificación de requisitos gracias a su legibilidad [82]. Esta forma de especificar es utilizada porque suele resultar más natural para los expertos. Este modelo se acerca mucho más al concepto de modelos de estados finitos que a las Redes de Petri.

Los Autómatas Finitos Deterministas y en general los modelos de estados finitos [91] pueden utilizar el concepto de estado como el de una actividad realizada en un momento dado y tras la realización de ésta utilizar los símbolos para escoger los siguientes estados a los que pasar, lo que permite un modelado más natural, legible y con menor complejidad gramatical que las Redes de Petri.

Sin embargo, los Autómatas Finitos Deterministas tienen un problema aun mayor que las Redes de Petri que solucionar: no son capaces de representar la concurrencia de ejecución de varias acciones al mismo tiempo, con lo que no son capaces de cumplir con los requisitos mínimos necesarios para la expresividad definida en los patrones de WF. El problema del paralelismo ha sido tratado en muchos artículos [58, 57, 70, 96] definiendo mejoras del Automata Finito Determinista para formar autómatas con capacidad de ejecutar actividades paralelas, sin aumentar la complejidad.

Definición 3.2 *Un Autómata Finito Paralelo (AFP)[96] M puede ser definido formalmente como una 7-tupla* $M = \{N, Q, \Sigma, \gamma, \delta, q_0, F\}$ *donde:*

N *es un conjunto finito de nodos,*
$Q \subseteq 2^N$ *es un conjunto finito de estados,*
Σ *es el alfabeto finito de entrada,*
$\gamma : 2^N \times (\Sigma \cup \{\lambda\}) \rightarrow 2^{2^N})$ *es la función de transición de los nodos,*
$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$ *es la función de transición de estados,*
$q_0 \in Q$ *es estado inicial,*
$F \subseteq N$ *es el conjunto de nodos finales.*

En la Figura 3.3 se puede ver el ejemplo de la Vía Clínica definida mediante una Red de Petri con una representación gráfica del AFP. Los nodos (N) son representados por los

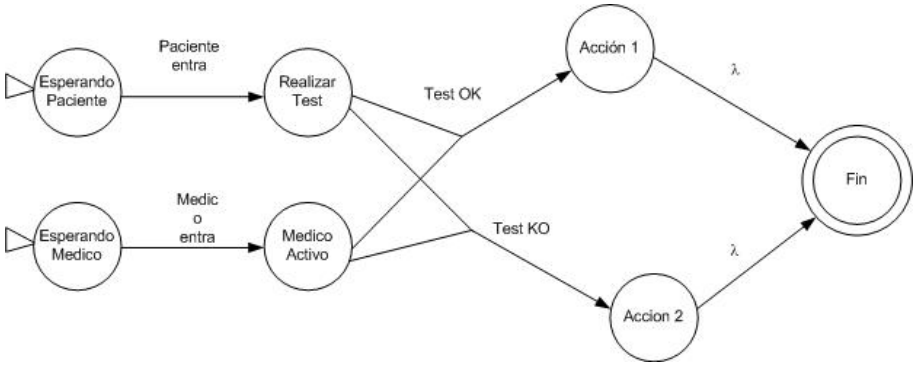

Figura 3.3: Ejemplo de Vía Clínica modelada con AFP

círculos y representan acciones a realizar, como *Realizar test* o *Acción 1*. Por otro lado, los estados (Q) representan grupos de acciones que pueden realizarse en paralelo, como *Realizar Test* y *Medico Activo*. Los estados pueden ser vistos como un marcado de los nodos activos en una Red de Petri. En el ejemplo, no existe un estado que aglutine los nodos *Accion 1* y *Accion 2* porque son incompatibles y nunca se ejecutarán en paralelo. La función γ representa la transición entre los nodos, por ejemplo, la transición entre *Realizar test* y *Medico Activo* y *Accion 1* es una transición múltiple que se representa como:

$$\gamma(\{RealizarTest, MedicoActivo\}, TestOk) \rightarrow \{Accion1\}$$

Por otro lado la función δ representa las relaciones entre los estados. Estas transiciones son uno a uno entre estados. Por ejemplo, la transición $\gamma(\{EsperandoPaciente\}, PacienteEntra) \rightarrow \{RealizarTest\}$ siempre se realiza en paralelo con los nodos *Esperando Medico* o *Medico Activo* de modo que desde el estado q_{EM_EP} que contiene los nodos *Esperando Medico* y *Esperando Paciente* según la función δ la transición sería:

$$\delta(q_{EM_EP}, PacienteEntra) \rightarrow q_{RT_EM}$$

Donde el estado q_{RT_EM} contiene a los nodos *Realizar Test* y *Esperando Medico*. Esta doble función permite la definición de patrones complejos, tales como patrones de sincronización o de ejecución paralela

Ventajas y limitaciones del AFP

Los AFP se adaptan mejor para el diseño de WF por parte de expertos, ya que están más cercanos a los modelos que tradicionalmente se utilizan para especificar gráficamente los procesos. Los modelos basados en Redes de Petri suelen tener gráficamente una talla mayor debido a la naturaleza de la formalización de la red. Esto hace que muchos expertos prefieran realizar el diseño de WF mediante la definición de Autómatas Finitos. En la Figura 3.3 se modela el mismo ejemplo que en la Figura 3.2 solo que en este caso se define con un AFP. Se puede ver claramente como el modelo basado en PFA requiere menos estructuras gráficas para ser descrito por lo que a la postre es más legible.

En cuanto a la expresividad del AFP, no existen estudios publicados que comparen estos modelos con los patrones de WF. Sin embargo, en [96], se realiza un estudio sobre las equivalencias entre el AFP y las Redes de Petri. En este estudio se llego a la conclusión de que cualquier Red de Petri segura, podría ser representada usando AFP, lo que asegura una buena expresividad. Por otro lado, la utilización de Redes de Petri seguras es suficiente para la realización de la mayoría de los problemas de WF existentes como se puede ver en [103].

Además, en el mismo trabajo [96], se demuestra que un AFP es equivalente a un Autómata Finito Determinista, con lo que es su complejidad esta acotada a la familia de los lenguajes regulares. Por otro lado, la complejidad gramatical de las Redes de Petri es muy alta, mucho más que la de los AFP [68]. Esto hace que la interpretación de los modelos basados en Redes de Petri sea mucho más difícil, como por otra parte ya avisaba Van der Aalst. Para más información, en [65] se hace un estudio formal sobre las Redes de Petri y los lenguajes regulares

Sin embargo, a pesar de que el AFP parece contar con una correcta expresividad, una mejor legibilidad para expertos que las Redes de Petri y una menor complejidad gramatical que estas, la principal y desalentadora desventaja de este modelo, es que no cuenta con una herramienta de modelado del tiempo, con lo que muchas de los problemas de WF, como las Vías Clínicas, estudiados no podrían modelarse con esta herramienta

3.1.3. Autómatas Temporales

Existen varios estudios publicados que tratan de modelar el tiempo en maquinas de estados finitas clásicas. En [11] se expone un completo estudio teórico sobre la idea del Autómata Temporal. En este trabajo se exponen tres modos de representar el tiempo:

- *Modelo de tiempo discreto*: en el que el tiempo se define como una secuencia de enteros monótona creciente.
- *Modelo de tiempo ficticio*: en el que el tiempo se define como una secuencia de enteros creciente.
- *Modelo de tiempo denso*: en el que el tiempo se define como una secuencia de numeros reales creciente.

Según [11], utilizar el modelado de tiempo denso complica la definición formal del modelo y aumenta la complejidad. Por su parte, la utilización de los modelos de tiempo discreto o ficticio, pueden ser fácilmente formalizados. Sin embargo, los modelos de tiempo discreto y ficticio perderían la visión natural del modelado del tiempo, ya que deben aproximar el tiempo escogiendo un quantum fijo.

Los Autómatas Temporales de tiempo denso definen relojes en las transiciones, y realizan operaciones sobre ellos. Un ejemplo de Autómata temporal se muestra en la Figura 3.4. En este ejemplo, en la transición desde el estado *S0* y el estado *S1* el reloj t se pone su valor a *0*, y la transición desde el estado *S1* al estado *S0* solo se habilitará si llega una palabra b en un tiempo inferior a *2* quantum de tiempo desde la ultima puesta cero del reloj. La definición formal del lenguaje temporal del ejemplo L_1 siendo $\Sigma = \{a, b\}$ es:

$$L_1 = \{((a,b)^\omega, \tau) | \forall i(\tau_{2i} < \tau 2i - 1 + 2)\}$$

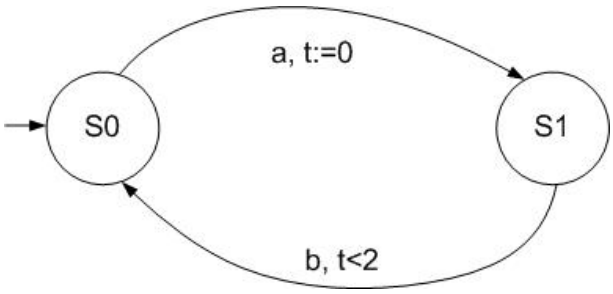

Figura 3.4: Ejemplo de Automata Temporal

Este estudio ha sido continuado y se han seguido aportando ideas y modificaciones sobre el problema. Analizando y mejorando la complejidad [36], formalizando herramientas de test [94], analizando en profundidad la complejidad algebraica del Automata Temporal [12] y proponiendo modificaciones sobre el modelo para aplicaciones específicas [95, 15]

Sin embargo, en cuanto a la expresividad necesaria para la definición de WF, no existen Autómatas Temporales preparados para realizar acciones paralelas por lo que su expresividad se ve bastante limitada. Esto es una clara limitación que hace poco viable la utilización de este tipo de modelos para la definición de WF para problemas como las Vías Clínicas.

3.2. Modelos Orientados a la Representación

Los modelos teóricos presentados anteriormente proponían un modelo formal para la descripción de modelos que siguen un patrón sintáctico formal de definición. Estos modelos fueron creados originalmente para la definición de procesos de un modo general. Sin embargo también se han definido modelos específicamente pensados para representar WF. Dentro estos están aquellos que han sido diseñados para ser fácilmente entendibles por expertos y que usualmente utilizan metáforas gráficas para describir la ejecución de los procesos. Estos modelos han sido tradicionalmente pensados para formar parte de las especificaciones de requisitos de los sistemas software [82] ya que ayudan a recoger el conocimiento del usuario de una forma fácilmente entendible y reproducible.

Al estar estos modelos tradicionalmente orientados a la especificación de los procesos de un modo únicamente descriptivo, pueden no contar con modelos de interpretación asociados. Sin embargo, es tal importancia que estos modelos han adquirido en el diseño de procesos que se han creado traductores para permitir la interpretación directa de estos WF.

En esta sección se van a analizar distintos sistemas orientados a la representación de WF.

3.2.1. BPMN

El lenguaje BPMN [59] (del inglés *Bussiness Process Modeling Notation*) es un estándar publicado por el Object Management Group. Este modelo es un sistema de notación gráfica para el dibujado de procesos de negocio. El objetivo principal de este modelo es la

Código	**Patrón**
WPC-1	Secuencia
WPC-2	Separación Paralela
WPC-3	Sincronización
WPC-4	Elección Exclusiva
WPC-5	Fusión Simple
WPC-6	Multielección
WPC-7	Fusión Sincronizada Estructurada
WPC-8	Multifusión
WPC-9	Discriminador Estructurado
WPC-10	Ciclos Arbitrarios
WPC-11	Terminación Implícita
WPC-12	Multiples instancias sin sincronización
WPC-13	Multiples instancias con conocimiento en tiempo de diseño
WPC-14	Multiples instancias con conocimiento en tiempo de ejecución
WPC-15	Multiples instancias sin conocimiento previo
WPC-16	Elección Aplazada
WPC-17	Rutina Paralela Entrelazada
WPC-18	Hito
WPC-19	Cancelación de Actividad
WPC-20	Cancelación de Instancia

Tabla 3.1: Listado de Patrones de control de WF usados para evaluar la expresividad de los modelos

creación de WF legibles por los diseñadores de procesos que les permitan definir de una forma sencilla sus procesos de negocio.

En cuanto a la expresividad, este lenguaje cumple bastantes de los patrones de WF de flujo de control definidos por Van der Aalst en [105] que pueden verse en la Tabla 3.1[1].Sin embargo, tiene problemas para expresar los patrones de discriminación estructurada (WPC-9), rutina paralela entrelazada (WPC-17) y los patrones de hito (WPC-18)

Aunque BPMN es un lenguaje estándar, no hay un formato de fichero estandarizado para almacenar modelos BPMN, por lo que el formato puede variar de unos sistemas a otros. Además, al no estar pensado para ser ejecutado, su interpretación no es directa, por lo que hay que realizar complejas labores de traducción para poder crear modelos de ejecución basados en BPMN

3.2.2. Diagramas de Actividad UML 2.0

Dentro de la iniciativa UML (del inglés *Universal Modeling Language*) también se han creado sistemas de notación gráfica para la representación de WF. Para ello, dentro de esta iniciativa, el Object Management Group ha adaptado el concepto de los diagramas de estado dentro de la versión 2.0 del standard para crear los llamados Diagramas de Actividad. Los Diagramas de Actividad [13], son utilizados para describir gráficamente secuencias de realización de actividades.

En cuanto a la expresividad, este modelo es ligeramente inferior a la del modelo BPMN, ya que además de los Patrones de WF que no es capaz de cumplir este, los Diagramas de Actividad de UML 2.0 no soportan la definición del patron de fusión sincronizada (WPC-7)

Al igual que BPMN a pesar de ser un lenguaje de representación estándar muy usado, no tiene un formato de fichero estándar que le dé la formalidad suficiente ni puede ser interpretado directamente.

3.2.3. XPDL

El lenguaje XPDL [24] (del inglés, *XML Process Definition Language*) es un estándar de la Workflow Management Coalition. Este modelo fue inicialmente diseñado para intercambiar el diseño de procesos de negocio entre distintas herramientas. En este lenguaje no solo se especifica la coordinación entre procesos, sino que también almacena el tamaño y las coordenadas X e Y de los items definidos.

En cuanto a la expresividad, al igual que el standard BPMN, cumple bastantes de los Patrones de WF de flujo de control definidos por Van der Aalst. Sin embargo, tiene problemas para expresar los patrones de discriminación estructurada (WPC-9), rutina paralela entrelazada (WPC-17) y los patrones de hito (WPC-18)

El lenguaje XPDL es un lenguaje que no solo solo provee de un conjunto de metáforas gráficas para el diseño de procesos, sino que también provee de un formato de fichero standard basado en XML. XPDL esta específicamente pensado para almacenar e intercambiar el diagramas de procesos. Sin embargo, XPDL no cuenta con un sistema estándar de Interpretación de WF. Algunos Sistemas de interpretación de WF como Enhydra Shark [99], Bonita [48] o WfMOpen [100] utilizan XPDL como lenguaje de representación de WF. La

[1]Para una descripción más exhaustiva de estos patrones se puede acudir al Anexo A

Patrón	BPMN	UML2.0	XPDL
WPC-1 (secuencia)	+	+	+
WPC-2 (separación paralela)	+	+	+
WPC-3 (Sincronización)	+	+	+
WPC-4 (Elección Exclusiva)	+	+	+
WPC-5 (Fusión Simple)	+	+	+
WPC-6 (MultiElección)	+	+	+
WPC-7 (Fusión Sincronizada Estructurada)	+	-	+
WPC-8 (MultiFusión)	+	+	+
WPC-9 (Discriminador Estructurado)	+/-	+/-	+/-
WPC-10 (Ciclos Arbitrarios)	+	+	+
WPC-16 (Elección Aplazada)	+	+	+
WPC-17 (Rutina Paralela Entrelazada)	-	-	-
WPC-18 (Hito)	-	-	-

Tabla 3.2: Evaluación de los patrones del control de flujo básicos en los modelos de representación estudiados

principal desventaja de XPDL es que no está pensado para ser ejecutado, con lo que es necesario añadir información específica no estándar al lenguaje que hace que los distintos sistemas de interpretación no sean compatibles entre si.

3.2.4. Evaluación de los modelos teóricos y orientados a la representación

En la Tabla 3.2 se presenta una comparativa de los modelos estudiados en cuanto a la expresividad[2]. En estas tablas únicamente están representados los patrones básicos[3] de flujo de datos estrictamente dependientes del lenguaje de representación, ya que los lenguajes de ejecución no son objeto de esta sección[4].

En esta tabla se puede ver como los lenguajes de descripción modelan casi todos los patrones de Van der Aalst. Únicamente patrones complejos como los de Hito, o Rutina Paralela Entrelazada, no pueden ser descritos por estos modelos. De los modelos orientados a la Representación es le lenguaje XPDL el que mejor resultado ofrece, ya no por su capacidad expresiva, sino por su formato de fichero estándar lo que facilita la creación de motores de interpretación asociados.

[2]El signo '+' representa que ese patrón puede ejecutarse en el modelo. El signo '-' representa que ese patrón no puede representarse en el modelo. El signo '+/-' representa que el patrón no puede representarse en el modelo, pero que puede simularse

[3]Los patrones de Van der Aalst son descritos en el Anexo A

[4]El patrón de Terminación Implícita (WPC11), los patrones de Control de Multiples Instancias (WPC-12 al WPC15) y los patrones de cancelación (WPC19 al WPC20) no son tomados en cuenta en cuanto a la representación de WF, ya que se refieren al manejo de las instancias en ejecución

3.3. Modelos Orientados a la interpretación de Flujos de Trabajo

Debido a las dificultades que añade la ejecución de modelos orientados a la representación, muchos investigadores han abordado el problema de la interpretación desde cero creando nuevos lenguajes específicamente pensados para ser interpretados. Estos lenguajes van a permitir una traducción mucho más directa desde la representación gráfica de los WF a la ejecución automática de estos mediante sistemas informáticos. Esta traducción directa hace la ejecución más eficiente, de hecho, algunos sistemas traducen directamente los procesos a código máquina directamente. Los modelos de interpretación de WF estudiados en esta Tesis son:

3.3.1. BPEL o WSBPEL

El lenguaje BPEL [53] (del inglés, *Business Process Execution Language*) es un lenguaje de orquestación de procesos definido por el comité de estandarización OASIS. BPEL nació como combinación de los lenguajes WSFL de IBM y XLANG de Microsoft. BPEL fue un acuerdo entre estas dos empresas para definir un lenguaje común que permitiera realizar conexiones entre las aplicaciones creadas tanto por productos de Microsoft como de IBM.

BPEL en un lenguaje eminentemente comercial serializado en XML cuya misión principal es definir procesos de negocio que interactúen con entidades externas a través de Web Services [32]. Por eso, BPEL es también conocido como WSBPEL. BPEL es un lenguaje pensado para ser ejecutado, por lo que no tiene una notación gráfica para el diseño de procesos. BPEL cuenta con la ventaja de la formalidad en la representación de la orquestación de procesos, ya que es un lenguaje que se ejecuta directamente por lo que no admite ambigüedad.

En cuanto a la capacidad de representación de los Patrones de WF, BPEL representa menos patrones que los modelos orientados a la Representación. Entre las limitaciones de BPEL en cuanto a la expresividad cabe destacar que no puede representar patrones de MultiFusión (WPC-8), es decir, no permite la fusión de ramas heterogéneas, patrones de Ciclos Arbitrarios (WPC-10), por lo que no permite la definición de ciclos de cualquier tipo y ni la definición de hitos (WPC-18), es decir, que impide representar transiciones dependientes de hitos alcanzados.

La principal ventaja de BPEL es que es un lenguaje estándar que se ha convertido en el lenguaje de referencia para hacer interactuar WebServices. Tanto es así que la mayoría de los motores del mercado tienen un módulo que les permite cargar módulos BPEL. Sin embargo, las llamadas nativas no están cubiertas por el modelo BPEL. Además la representación adolece de patrones importantes como el de ciclos arbitrarios, que es un patrón muy común en los sistemas de WF.

3.3.2. jBPM

jBPM [43] (del inglés, java Business Process Management) es un motor de WF implementado en lenguaje Java que funciona sobre servidores JBoss sobre la plataforma J2EE. jBPM es una solución open source para proveer de un conjunto de herramientas para la gestión e interpretación de WF. Este modelo incluye, además de un lenguaje formal basa-

do en BPEL llamado jPDL(java Process Description Languaje), una herramienta gráfica de diseño de WF que permite fácilmente la definición de procesos de negocio arrastrando y soltando primitivas de WF.

En cuanto a la capacidad de representación, jPDL es un lenguaje basado en BPEL por lo que es bastante parecido. Sin embargo, las ultimas versiones de jBPM han solucionado algunos de los problemas que tiene BPEL con la expresividad de los ciclos arbitrarios.

jBPM tiene implementado un motor de ejecución capaz de ejecutar los WF diseñados tanto por la herramienta gráfica, como por otras herramientas capaces de generar código en BPEL. jBPM está orientado a su utilización como orquestador de Web Services, aunque puede trabajar también como orquestador de procesos codificados en Java. Debido a la necesidad de una ejecución eficiente, el equipo de desarrollo de jBPM en su implementación actual, ha optado por la creación de un traductor de WF a código nativo para la ejecución. Esto, que hace que el código sea mucho más eficiente que la interpretación directa de los WF, pero hace que no sea posible la modificación de WF en tiempo dinámico. Esto es un problema para muchos de las aplicaciones estudiadas, por ejemplo, en el caso de las Vías Clínicas, un alto porcentaje de los WF ejecutados cambian a lo largo de su vida, debido, sobre todo, a las usuales pluripatologías de los pacientes. Usando jBPM, la única solución posible es cancelar el WF actual y ejecutar otro nuevo.

3.3.3. Windows Workflow Foundation

El modelo WWF [34](del inglés, *Windows Workflow Foundation*) es el sistema propuesto por Microsoft para la gestión de WF. En su filosofía de creación de un marco común para todas sus aplicaciones, Microsoft ha creado el WWF como un conjunto de utilidades compatibles con los lenguajes de programación enmarcados dentro de lo que se denomina como plataforma .NET [81]. Este modelo es un sistema propietario inicialmente basado en el standard BPEL sobre el que se le han aplicado ciertas mejoras basadas en el estudio de las necesidades de los usuarios.

En cuanto a la capacidad de representación de patrones de WF, WWF es capaz de expresar prácticamente los mismos patrones de WF que BPEL, exceptuando, que gracias a la capacidad de definir procesos basados en estados, puede generar ciclos arbitrarios. Sin embargo, debido a que no se pueden mezclar en el mismo WF modelos de estados y modelos de flujo, no siempre es posible ejecutar todas las combinaciones de patrones posibles.

Al igual que jBPM, este sistema incorpora una herramienta gráfica que permite el diseño de WF. Sin embargo, WWF es menos compatible con BPEL, ya que incorpora nuevas mejoras propias sobre el lenguaje. Una de estas mejoras es la capacidad que tiene WWF de diseñar WF desde un punto de vista de estado, en lugar del clásico flujo de datos definido por BPEL.

WWF tiene un eficiente motor de ejecución capaz de ejecutar tanto WF compilados, como código BPEL standard. Al contrario de jBPM, WWF es capaz de modificar instancias de WF dinamicamente, lo que lo abre un amplio abanico de posibilidades. Sin embargo, modificar un WF en ejecución es una ardua tarea que solo puede ser ejecutada por un programador o debe ser automatizada por herramientas externas específicamente creadas a tal efecto, cosa que limita su aplicabilidad.

3.3.4. Staffware

Staffware Process Suite [64] es una aproximación diferente a la de BPEL. Staffware es un producto clásico en la gestión de procesos de negocio. Este modelo es un paquete de aplicaciones pensado para gestionar los flujos de control de las empresas en las que se encuentra instalado. Staffware es un sistema propietario cerrado difícil de personalizar. Este sistema, maneja los procesos de una forma holística de modo que todos los procesos que están a su control pueden interconectarse, permitiendo la incorporación de modelos de coreografía de procesos.

En cuanto a su capacidad de representación cabe destacar la capacidad que tiene la implementación del motor de ejecución de Staffware para la gestión de la interconexión de multiples instancias de WF, lo que le valida para la ejecución de sistemas basados en coreografía de procesos. Por otro lado, carece de la capacidad de representar patrones de flujo básico como puedan ser Hitos (WPC-18), rutinas paralelas entrelazadas (WPC-17), MultiElección (WPC-6) o Fusión Sincronizada estructurada (WPC-7) lo que le inhabilita para representar los procesos que contengan este tipo determinado de flujos.

Staffware contiene herramientas para definir gráficamente procesos, para la automatización y seguimientos de WF, para la modificación dinámica de los procesos, así como la monitorización, planificación y gestión de los recursos de los recursos y actores involucrados en los procesos. La principal desventaja de Staffware es que es un lenguaje propietario que es difícil de personalizar por lo que los procesos en ejecución han de amoldarse al sistema y no al revés.

3.3.5. Evaluación de los modelos orientados a la interpretación

En la Tabla 3.3 se presenta la comparativa de los modelos basados en la interpretación de WF estudiados en cuanto a la expresividad[5]

Los modelos orientados a la interpretación por su naturaleza formal usualmente suelen tener recortadas sus posibilidades expresivas. A pesar de usar metáforas gráficas para mejorar la legibilidad de los modelos, estas son en muchas ocasiones demasiado próximas a los lenguajes imperativos, lo que en casos como las Vías Clínicas hacen más dificultosa la legibilidad ya que los facultativos no suelen estar acostumbrados a este tipo de notaciones. Algunos modelos como el WWF intentan mejorar estos problemas proponiendo varias formas de diseñar los modelos. Entre ellas, usando modelos basados en estados que son mucho más intuitivos para los expertos. Sin embargo, esta vista de diseño en el WWF es poco expresiva, ya que no permite la ejecución de estados paralelos.

3.4. Representación e Interpretación de Vías Clínicas basadas en Modelos de Flujos de Trabajo

En esta sección se han analizado distintos modelos de Representación e Interpretación de WF en cuanto a su capacidad expresiva, su legibilidad, y su complejidad de ejecución.

[5]El signo '+' representa que ese patrón puede ejecutarse en el modelo. El signo '-' representa que ese patrón no puede representarse en el modelo. El signo '+/-' representa que el patrón no puede representarse en el modelo, pero que puede simularse

Patrón	BPEL	jBPM	WWF	Staffware	FLOWer	YAWL
WPC-1 (secuencia)	+	+	+	+	+	+
WPC-2 (separación paralela)	+	+	+	+	+	+
WPC-3 (Sincronización)	+	+	+	+	+	+
WPC-4 (Elección Exclusiva)	+	+	+	+	+	+
WPC-5 (Fusión Simple)	+	+	+	+	+	+
WPC-6 (MultiElección)	+	+	+	-	+	+
WPC-7 (Fusión Sincronizada Estructurada)	+	+	+	-	+	+
WPC-8 (MultiFusión)	-	-	-	-	+/-	+
WPC-9 (Discriminador Estructurado)	-	-	-	-	-	+
WPC-10 (Ciclos Arbitrarios)	-	-	+/-	+	-	+
WPC-11 (Terminación Implicita)	+	+	+	+	+	+
WPC-12 (MI sin Sincronización)	+	+	+	+	+	+
WPC-13 (MI con conocimiento en Diseño)	-	-	-	+	+	+
WPC-14 (MI con conocimiento en Ejecución)	-	-	-	+	+	+
WPC-15 (MI sin conocimiento Previo)	-	-	-	-	+	+/-
WPC-16 (Elección Aplazada)	+	+	+	-	+	+
WPC-17 (Rutina Paralela Entrelazada)	+/-	+/-	+/-	-	+/-	+
WPC-18 (Hito)	-	-	-	-	+/-	+
WPC-19 (Cancelar Actividad)	+	+	+	+	+/-	+
WPC-20 (Cancelar Instancia)	+	+	+	-	+/-	+

Tabla 3.3: Evaluación de los patrones del control de flujo básicos en los modelos de interpretación estudiados

Las Vías Clínicas requieren que los modelos de representación sean muy expresivos. Esto es debido a que el cuidado las enfermedades en la realidad coordinan complejos tratamientos que se traducen en complejos patrones de ejecución. En esta línea los modelos teóricos como las Redes de Petri o los AFP, son más adecuados porque son capaces de representar más patrones. Sin embargo, los AFP no son capaces de representar el tiempo lo que resulta una gran limitación en la Vías Clínicas.

En cuanto a la legibilidad, los expertos en medicina suelen preferir los modelos basados en metaforas, frente a los modelos basados en lenguajes de programación, ya que les resultan más intuitivos. En este caso, los modelos orientados a la representación son los más adecuados para el diseño.

En cuanto a la complejidad gramatical, los modelos de representación son ambiguos, muy complejos y difíciles de ejecutar. Los modelos orientados a la interpretación son los más cercanos a los lenguajes imperativos y son más sencillos de ejecutar. Por último, en cuanto a los modelos teóricos, por un lado las Redes de Petri serían complicadas de ejecutar debido a su alta complejidad gramatical, y los modelos de estados finitos, serían fáciles de ejecutar debido a su baja complejidad gramatical.

Por otro lado, estudios realizados con expertos en el proyecto CAREPATHS [6] demuestran que los diseñadores de Vías Clínicas requieren modelos altamente expresivos y que permitan ser modificados dinámicamente, debido a la complejidad del tratamiento de los pacientes. En esta línea, los modelos orientados a la interpretación tienen problemas en cuanto a expresividad, y no son fáciles de ser modificados en tiempo de ejecución. Los modelos orientados a la representación, son más expresivos y legibles, pero debido a la necesidad de una traducción entre el lenguaje de representación su modificación dinámica no es un tema baladí. Por ultimo, los modelos teóricos, son los más expresivos y tienen una legibilidad aceptable. En el mismo proyecto CAREPATHS, los expertos mostraron predilección por los modelos basados en estados frente a la utilización de Redes de Petri de alto nivel.

Como conclusión, la mejor opción para la creación de un modelo de representación de WF para Vías Clínicas, sería la utilización de un modelo de estados finitos paralelo con capacidad de modelar el tiempo, ya que tendría una buena expresividad, y una legibilidad adecuada en conjunción con una complejidad aceptable y en consonancia con las opiniones de los expertos.

Capítulo 4

Aprendizaje de Flujos de Trabajo

Tras haber estudiados modelos de representación e interpretación de WF, en este capítulo se va a analizar una técnica para aprender WF a partir de muestras anteriores que permite apoyar el diseño de WF. Además se analizará el estado del arte en este tipo de técnicas.

4.1. Introducción

A pesar de los grandes esfuerzos en la representación de los procesos de negocio, el diseño por parte de expertos encuentra muchas dificultades añadidas. Un ejemplo de esto es el problema de la gestión de Vías Clínicas. Los procesos de Vías Clínicas son muy difíciles de definir, y mucho más de construir formalmente. Para la realización de esta estandarización de procesos es necesaria la coordinación de grupos de expertos en la materia para el consenso de protocolos de actuación para las tareas. Aun así, estos protocolos no están exentos de errores debido a la subjetividad que aporta el diseño a través de expertos haciendo aún más difícil la adecuación de estos procesos a la realidad. Por esto, son necesarias muchas iteraciones de implantación del proceso de estandarización de procesos para conseguir resultados esperanzadores. Este tedioso proceso, muchas veces supone una barrera que impide la implantación de las Vías Clínicas. Además, las Vias Clínicas pueden depender de los hábitos de población, por lo que puede ser que procesos que funcionen con un grupo de personas, no funcionen con otros.

La alta variabilidad de los pacientes en el proceso de cuidado hace que no todos los tratamientos funcionen en todos los casos. Un caso claro de esto son los pacientes pluripatológicos. Este tipo de pacientes pueden tener complicaciones añadidas que hagan modificar las Vías Clínicas inicialmente diseñadas en pos de adecuar los cuidados a las necesidades de este. Por ejemplo, en un paciente cuyo tratamiento, según la Vía Clínica, recomiende una operación pero que presente problemas de coagulación requiere modificar la instancia de ejecución por problemas de incompatibilidad. Estas modificaciones solo han de hacerse en la instancia, y no en la plantilla para así no afectar a futuros pacientes.

Estas excepciones de la Vía Clínica en tiempo de ejecución resultan muy comunes en la vida real. En algunos casos, se deben a pacientes excepcionales, cuyos cuidados raramente se repetirán, pero en muchas ocasiones estas modificaciones se repiten a menudo y pueden añadirse al modelo para que este sea más fiable. En este caso, La incorporación de estos datos al modelo necesita de complejos estudios de las ejecuciones pasadas para detectar

estas modificaciones e incorporarlas al modelo.

La utilización de técnicas de reconocimiento de formas para la creación de WF que representen la realidad puede ser la solución a los problemas de estandarización de este tipo de procesos. La inferencia de WF a partir de corpus de instancias de WF pasados puede permitir a los diseñadores de procesos obtener vistas de alto nivel sobre las ejecuciones de los procesos con una menor subjetividad y basados en la ejecución real de dichos procesos.

4.2. Aprendizaje de Flujos de Trabajo

El Aprendizaje de Flujos de Trabajo (WM, del ingles *Workflow Mining* o *Process Mining*) es una técnica para la inferencia de procesos de negocio basada en el aprendizaje automático de WF a partir de instancias de ejecución. El objetivo del WM es darle la vuelta al proceso, y a modo de reingeniería, recoger datos en tiempo de ejecución (instancias de WF) para aportar información a la hora del análisis y diseño de WF [106].

La tarea de diseño de WF parece la aplicación más clara del WM. Sin embargo, su aplicación directa no siempre es posible. Uno de los principales problemas que se pueden encontrar al diseñar un WF para resolver un determinado problema mediante WM, es que no haya un WF previo en ejecución que inferir. Esto exige que el diseño del WF se haga manualmente, con los problemas de diseño que esto conlleva.

Para solucionar esto, es posible realizar el diseño del WF de un modo iterativo mediante un modelo en espiral, implantando un WF general inicial e ir mejorándolo mediante la realización de reingeniería de procesos en sucesivas iteraciones. Este primer WF suele estar bastante alejado de la realidad, lo que obliga a las instancias ejecutadas a ser continuamente modificadas en su ejecución. Los caminos que siguen estas instancias, pueden ser utilizados para inferir un nuevo WF que sea más acorde con los procesos reales que el inicialmente diseñado. Estos flujos inferidos son presentados a los diseñadores de procesos de negocio para que medie en el proceso de diseño solucionando manualmente los posibles errores de inferencia.

En cuanto a su aplicación a las Vías Clínicas, estas técnicas permitirían inferir los protocolos de cuidado generales a partir de la practica diaria y de los facultativos. De este modo la modificación de las Vías Clínicas en la práctica diaria pueden ser usada para inferir modelos adaptados a la realidad de la ejecución de los procesos de cuidado. Gracias a esto los expertos definirán con una mayor facilidad procesos estandarizados con una mayor probabilidad de ser más aceptados en la práctica diaria.

4.2.1. Trazas de Flujos de Trabajo

Los sistemas Software en general y los Sistemas de Interpretación de WF en particular suelen utilizar archivos donde van añadiendo la información relativa a la ejecución de los procesos para registrar las movimientos y actividades durante un tiempo determinado. Estos archivos se llaman trazas (del inglés *Logs*) y se almacenan usualmente en ficheros con formato de texto plano escrito en standard ASCII.

En el caso de los sistemas de Interpretación de WF, los ficheros de Traza de Flujos de Trabajo (WLogs, del inglés *Workflow Logs*) se encargan de almacenar todos los eventos producidos por los WF en funcionamiento, almacenando la fecha y hora, el evento ocurrido,

y la instancia que ha producido el evento. Esta información puede ser usada para crear corpus que pueden ser utilizados por los sistemas de WM para inferir WF.

4.3. Aplicaciones del Aprendizaje de Flujos de Trabajo

En esta sección se van a describir aplicaciones donde el WM está demostrando su utilidad en la actualidad. Entre estas aplicaciones se encuentran el diseño de WF, el aprendizaje de buenas prácticas, el aprendizaje en línea de procesos eficientes o la reingeniería de procesos

4.3.1. Aprendizaje de buenas prácticas

El diseño de WF orientado a la mejora de los procesos existentes en una empresa, es usualmente dependiente de las características específicas de ésta. Sin embargo, es posible que la información aprendida de los procesos inferidos pueda llevar a un conocimiento general que permita ser trasladado a procesos similares existentes en otros entornos.

Estos WF generales aprendidos pueden servir de punto de partida para la el diseño de WF más específicos basados en experiencias positivas probadas en entornos similares. Este sistema puede acortar el numero de iteraciones necesarias pera encontrar el WF más eficiente posible, lo que aumenta las probabilidad de aceptación de los procesos implantados.

4.3.2. Aprendizaje en linea de procesos eficientes

En muchas ocasiones, la utilización de sistemas de apoyo a la decisión, es muy positiva a la hora de gestionar procesos. La toma de decisiones en cada una de las ramas de un WF puede provocar que una instancia finalice correctamente o no. La creación de sistemas expertos para facilitar la decisión en las ramas de un WF puede requerir mucho tiempo de diseño, y se ve afectado por los continuos rediseños del WF en ejecución.

Una técnica para solucionar este problema es la utilización del WM para inferir WF basados en historias pasadas donde utilizando muestras similares a la que está en ejecución podamos predecir el comportamiento del WF dependiendo de la necesidad a tomar. De este modo las instancias bien acabadas nos mostrarán las decisiones que, en ese punto han tenido éxito en procesos similares. Por otro lado, las instancias mal acabadas nos mostrarán las decisiones que provocaron caminos erróneos en el sistema. Con esta información, los responsables de los procesos contarán con una mayor probabilidad de acertar en sus decisiones.

4.3.3. Reingeniería de Procesos

Desde que en 1990 Hammer publicara en [62] su visión de la ingeniería de procesos en las organizaciones, las empresas han intentado mejorar su productividad mediante la reingeniería de procesos. La reingeniería de procesos se basa en el analisis de los procesos actualmente en ejecución en las empresas, para la mejora de estos. Esto tradicionalmente requiere un profundo estudio, usualmente realizado por expertos externos, que auditan los métodos de actuación de las empresas. Para que este modelo sea más efectivo se requiere

Id Instancia	Evento
Instancia 1	Tarea A
Instancia 2	Tarea A
Instancia 3	Tarea A
Instancia 3	Tarea B
Instancia 1	Tarea B
Instancia 1	Tarea C
Instancia 2	Tarea C
Instancia 4	Tarea A
Instancia 2	Tarea B
Instancia 2	Tarea D
Instancia 5	Tarea A
Instancia 4	Tarea C
Instancia 1	Tarea D
Instancia 3	Tarea C
Instancia 3	Tarea D
Instancia 4	Tarea B
Instancia 5	Tarea E
Instancia 5	Tarea D
Instancia 4	Tarea D

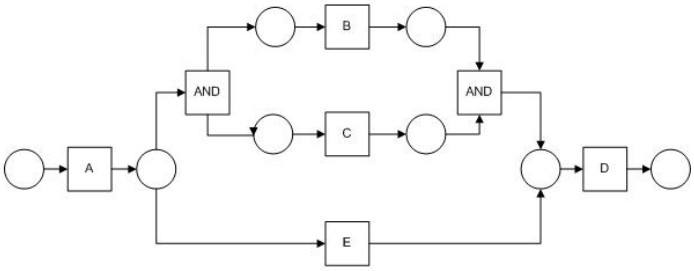

Figura 4.1: Ejemplo de Inferencia de Workflows Basado en Eventos

un vasto conocimiento de los procesos en ejecución de las organizaciones que normalmente es complejo de conseguir y requiere mucho esfuerzo por parte de la empresa. Utilizando técnicas de WM es posible conocer una visión real de los flujos de la empresa ya que nos permite ver a un alto nivel, como los procesos y acciones van ejecutándose, permitiendo una fácil localización de los puntos más críticos para modificarlos en pos de mejorar la productividad del flujo general de la empresa.

4.4. Aprendizaje de Flujos de Trabajo Basado en Eventos

El mayor problema a la hora de inferir un modelo de WF a partir de instancias de ejecución es el número de muestras que podemos utilizar para el aprendizaje. Este es un gran problema que en muchos casos tiene difícil solución. Debido a este problema la comunidad científica ha optado por utilizar el WM en los entornos que más muestras sean capaces de generar utilizando la información disponible en los WLogs. Dentro de la disciplina del WM, dada la gran cantidad de muestras disponibles, el ámbito más estudiado es el aprendizaje de Flujos de Trabajo Basado en Eventos.

El Aprendizaje de Flujos de Trabajo Basado en Eventos [104], (EBWM, del inglés *Event-Based Workflow Mining*) es una disciplina que utiliza el conjunto de eventos ocurridos durante toda la vida de cada una de las instancias de Workflow para formar un corpus que permita la inferencia de WF.

En otras palabras, los algoritmos de EBWM utilizan como entrada corpus formados por instancias de WF donde en cada instancia se especifica ordenadamente todos los eventos que han ocurrido en dicha instancia durante en ciclo de vida esta instancia. Gracias a la facilidad con la que se encuentran estos datos en los WLogs del mercado, este tipo de WM ha alcanzado tal importancia que ha pasado a ser prácticamente la única técnica investigada en esta relativamente nueva disciplina.

En la Figura 4.1 se presenta un ejemplo de EBWM. En la tabla de la Figura se

muestran cronológicamente ordenadas las tareas que se han ido ejecutando en cada una de las instancias que forman el corpus. En la columna de la izquierda se ve la instancia que ha generado el evento, mientras que en la columna de la derecha se muestran los evento que se han generado en esa instancia. Cada una de las instancias (5 en el ejemplo) representa un camino en el WF. Los algoritmos de EBWM utilizan esta información para inferir modelos completos que describen la ejecución de las instancias. El modelo representado en la Figura, representado en forma de Red de Petri se puede ver el WF inferido que acepta las muestras de entrada.

4.4.1. Algoritmos de aprendizaje de Flujos de Trabajo basado en Eventos

En la literatura existen diversos algoritmos de EBWM. En [33] se presenta uno de los primeros trabajos en esta disciplina, en los que se presentan tres métodos de inferencia gramatical para aprender gramáticas regulares que modelen el sistema. Mas adelante, Will Van der Aalst publica su primer algoritmo de WM: el α Algorithm [104] este algoritmo está basado en la inferencia de Redes de Petri a partir de WLogs. Este algoritmo fue extendido posteriormente con el $\alpha{+}{+}$ Algorithm [40]. La principal novedad de estos algoritmos es que ya intentan inferir las estructuras paralelas utilizando las Redes de Petri. Otros algoritmos utilizan nuevos tipos de estructuras para inferir procesos. Por ejemplo, el Multi-Phase Process Mining [107] infiere unas estructuras llamadas Cadenas de Procesos Dirigidas por Eventos [67] más orientadas al guiado de procesos que para mostrar el modo de funcionamiento un flujo de procesos, por lo que pecan de perdida de legibilidad.

Ya en pleno auge de los patrones de WF se presentó un trabajo orientado a aprender directamente modelos que incorporaran como primitivas estos patrones como el Workflow Patterns Miner [54]. También técnicas relativamente novedosas como los algoritmos genéticos han sido utilizados para inferir WF como es el caso de el algoritmo Genetic Process Mining [41]. Otro algoritmo a destacar es el Heuristic Miner [109] este algoritmo está orientado a inferir WF resistentes al ruido.

Muchos de estos algoritmos se encuentran disponibles en el paquete ProM [108]. El ProM es una herramienta software de código abierto escrita en JAVA que proporciona un marco de programación de algoritmos de EBWM. ProM propone una estructura XML para definir los corpus de datos en un formato común para que todos los algoritmos implementados en ProM puedan hacer uso de éstos.

En esta Tesis se han estudiado en mayor detalle los cuatro algoritmos más conocidos. Estos algoritmos serán utilizados en los experimentos como método de contraste con el algoritmo propuesto en esta Tesis que se detallará más adelante. Estos algoritmos se encuentran implementados dentro del paquete ProM y producen un WF en lenguaje Dot. Estos algoritmos son el Heuristic Miner, el Genetic Process Miner, el α y el $\alpha{+}{+}$:

Algoritmo Heuristics Miner

El algoritmo Heuristic Miner presentado en [109] esta basado en la construcción de un grafo de dependencias a partir los distintos eventos que se producen dentro de un WLog. Este grado de dependencia a la postre representará el WF inferido por el sistema. El algoritmo Heuristic Miner solo considera el orden de los eventos en la instancia.

El grafo de dependencias contendrá como nodos, los eventos ocurridos en el WLog. Los arcos se calculan mediante la función $a \Rightarrow_w b$, siendo a y b eventos del WLog. La función se defina formalmente como:

$$a \Rightarrow_w b = \frac{|a >_w b| - |b >_w a|}{|a >_w b| + |b >_w a| + 1}$$

La función $a >_w b$ representa si el evento b se alcanza directamente desde a con únicamente un paso. $|a >_w b|$ representa el numero de veces que tras un evento a ocurre b en la traza de la instancia.

Dados a y b, $a \Rightarrow_w b$ es un numero entre -1 y 1. Cuanto más cercano a -1 menor probabilidad de ser una dependencia, mientras que cuanto más cercano a 1 mayor probabilidad de serlo. Por otro lado, si es cero, significa no existen datos de esa relación. Si el resultado de esta función es mayor que un umbral dado, se dibujará un arco entre a y b. Los eventos y sus arcos asociados formarán, finalmente, el WF inferido.

Este algoritmo construye WF muy sencillos. Sin embargo, este algoritmo, está pensado para inferir WF con patrones sin acciones paralelas ni sincronizaciones. Esto es debido a que las características de los grafos impiden la definición de procesos paralelos. Aunque en el artículo del Heuristic Miner [109] se habla de extensiones del grafo de dependencias para resolver este problema, no existen implementaciones reales de esto en el paquete ProM.

Algoritmo Genetic Process Mining

Los algoritmos genéticos son algoritmos que imitan los procesos biológicos de la evolución para encontrar modelos de comportamiento. Los algoritmos genéticos son procesos iterativos en los que las hipótesis van mutando, y combinandose con respecto a una medida de bondad que marcara si el modelo evolucionado ha mejorado o empeorado. Cada nueva hipótesis generada es llamada nueva generación. Las iteraciones se suceden hasta que un criterio de parada de produce, que normalmente viene dado por un criterio de bondad alcanzado o numero de iteraciones máximo.

El *Genetic Process Mining*, presentado en [41], sigue este modelo tomando como población de hipótesis WFs que se combinan y mutan pseudoaleatoriamente para mejorar en cada iteración. Estas dos operaciones se efectúan sobre una generación para evolucionar hacia otra generación. La operación de combinación mezcla de una manera pseudoaleatoria dos hipótesis para evolucionar otra nueva hipótesis. Por otro lado la operación de mutación añade, también de modo pseudoaleatorio, eventos a una hipótesis para generar una nueva generación. Las mejores evoluciones generadas serán las hipótesis que existan en la siguiente generación.

Para comprobar si la hipótesis de nueva generación es mejor o peor que la anterior se utiliza como modelo de bondad la siguiente función:

$$Bondad(i) = Exactitud(i) - k * Precision(i)$$

La medida de bondad se basa en dos conceptos, la Exactitud y la Precision.

La Exactitud es una medida de como la hipótesis acepta las muestras de entrada. cuanto mayor sea la Exactitud, mejor será la bondad del modelo. La Exactitud se calcula utilizando la siguiente formula:

$$Exactitud = \frac{E_C - \frac{E_F}{T_T - T_F + 1} - \frac{E_E}{T_T - T_F + 1}}{E_T}$$

Siendo E_C el número de eventos aceptados por la hipótesis, E_F el número de eventos no aceptados por la hipótesis, E_E el número de eventos extra definidos en la hipótesis e inexistentes en la muestra, T_T el número de muestras totales y T_F el número de trazas que han fallado.

Por otro lado, la Precisión mide la información extra que se infiere en la hipótesis. Cuanto mayor sea la Precisión peor será la bondad del modelo. La precisión se calcula utilizando la siguiente formula:

$$Precision(i) = \frac{EventosAccedidos_i}{max(EventosAccedidos_{1..n})}$$

Donde $EventosAccedidos_i$ son el numero de nodos que han sido accedidos durante el proceso de verificación de la instancia i de la muestra.

Al igual que el Heuristic Miner, el algoritmo de Genetic Process Mining utilizado (proporcionado por ProM) genera un grafo de dependencias que representa el WF, lo que facilita le creación de WF legibles. Esto resulta una limitación ya que no es posible representar patrones paralelos o de sincronización utilizando estos modelos. Aunque en el artículo del Genetic Process Mining [41] se habla de extensiones de este algoritmo para inferir Redes de Petri, no existen implementaciones reales de esto en el paquete ProM.

Algoritmo α

El Algoritmo α, presentado en [104], es uno de los primeros algoritmos de EBWM que está específicamente dedicado a inferir WLogs obteniendo como resultado Redes de Petri.

El algoritmo α primero examina las trazas para crear el conjunto de eventos (*Transitions*) diferenciando las de entrada las de salida. A continuación se seleccionan los eventos y se ordenan según su relación causal. El algoritmo primero selecciona todos los eventos, y va refinando las secuencias primando los eventos más largos. Al final de las iteraciones se consigue finalmente la Red de Petri resultante.

El algoritmo α es capaz de generar Redes de Petri a partir de WLogs. Esto le permite la inferencia de procesos con patrones más complejos, como rutinas paralelas y sincronizaciones. Sin embargo, por la naturaleza de las Redes de Petri, los modelos tienden a ser menos legibles

Algoritmo α ++ Algorithm

El algoritmo $\alpha++$, presentado en [40], nació como una modificación del algoritmo α debido a sus debilidades a la hora de inferir WF con ciclos simples. Del mismo modo que el algoritmo α, este algoritmo también es capaz de inferir Redes de Petri a partir de WLogs.

El algoritmo $\alpha++$ primero examina las trazas e identifica los ciclos de longitud uno (ciclos simples), que son registrados y apartados del corpus. A continuación se aplica el algoritmo α al WF. Posteriormente se vuelven a colocar los ciclos problemáticos en sus lugares correspondientes dando lugar a la Red de Petri resultante.

Al igual que el algortimo α, El algoritmo $\alpha++$ es capaz de generar Redes de Petri a partir de los WLogs, lo que permite la inferencia de procesos con patrones más complejos, como rutinas paralelas y sincronizaciones. Sin embargo, por la naturaleza de las Redes de Petri, los modelos tienden a ser menos legibles

4.5. Aprendizaje de Flujos de Trabajo Basado en Eventos y Vías Clínicas

Como ya se ha explicado, los modelos de WM basados en eventos, utilizan sólo junto al nombre de la tarea la información temporal de inicio y a veces de fin de las tareas realizadas para inferir los WF. Este modelo, sirve para descubrir acciones que repetidamente se producen tras la realización de otras y que podrían ser incorporadas al modelo original, o que siguen un patron se sincronización con los demás procesos en funcionamiento.

Sin embargo, en muchas ocasiones, las Vías Clínicas varían su ejecución basándose en resultados de las acciones anteriores. Por ejemplo, cuando una persona se toma la temperatura, dependiendo del resultado de la medida (Fiebre o temperatura normal) puede requerir la toma de antipiréticos como el paracetamol o no tomarse nada. La información utilizada por los algoritmos de WM basada en eventos es insuficiente para inferir esta causalidad, por lo que no cubren las expectativas para el apoyo al diseño de las Vías Clínicas. Por tanto si tenemos como objetivo la utilización de técnicas de WM para apoyar el diseño de Vías Clínicas, es necesario **enriquecer los corpus** con información sobre el resultado de las acciones, y **crear nuevos algoritmos** que sean capaces de aprovechar dicha información para crear WF con mayor información sobre la ejecución de los procesos.

Por otro lado, en el campo del aprendizaje, la complejidad gramatical del modelo de representación cobra especial importancia, ya que cuanto mayor sea la complejidad gramatical, mayores dificultades encontramos en la inferencia. Por ello, cuanto menor sea la complejidad, mejor funcionará el algoritmo de inferencia. Por ello, para facilitar el aprendizaje hay que **seleccionar un lenguaje de representación de WF con la complejidad gramatical lo más sencilla posible**. En este caso los modelos basados en estados finitos suponen una buena elección para facilitar el aprendizaje.

En este trabajo se va a proponer un nuevo paradigma alternativo al EBWM, que supone el enriquecimiento de los corpus de entrenamiento con datos referentes a los resultados de los procesos en ejecución. Basado en esta nueva metodología se presentará un nuevo algoritmo de WM capaz de aprovechar esta información para inferir la causalidad de los procesos en ejecución plasmándola formalmente en un lenguaje de WF. Este algoritmo será útil para ayudar a los expertos en Vías Clínicas en su diseño.

Además en esta Tesis se va a proponer un modelo de representación de WF con una baja complejidad gramatical, manteniendo una alta expresividad y legibilidad, que facilite la inferencia de los procesos.

Además, se va presentar un algoritmo basado en este paradigma capaz de inferir WF representados utilizando modelos con la menor complejidad gramatical posible pero con la mayor expresividad y legibilidad posible.

Parte II

Flujos de Trabajo basados en Actividades

Capítulo 5

Aprendizaje de Flujos de Trabajo basado en Actividades

Hasta este momento se ha abordado la realización de un estudio de los modelos actuales de Representación, Interpretación y Aprendizaje de Flujos de Trabajo. A partir de este capítulo se van a realizar propuestas en cada uno de estos ámbitos para afrontar el problema del diseño de WF para su aplicación a las Vías Clínicas.

En este capítulo se va a proponer una alternativa al EBWM capaz de abordar los problemas que quedan fuera de esta metodología. Para definir esta nueva metodología se acotaran los datos mínimos necesarios y se identificarán entornos donde este sistema obtiene ventaja sobre el EBWM. Pensando en esta metodología se plantearán nuevas herramientas y algoritmos que pueden ser útiles para apoyar su implantación

5.1. Limitaciones actuales en el Aprendizaje de Flujos de Trabajo

Las técnicas de EBWM están basadas en los datos que ofrecen los sistemas de interpretación actualmente. En los WLogs que se pueden obtener de estos sistemas solo hay información de los eventos ocurridos durante la ejecución de los WF. Esta información se reduce a los datos temporales de los eventos de inicio y de fin de las acciones. Con esta información los sistemas de EBWM son capaces de inferir WF donde se descubren las secuencias de acciones que se siguen en diferentes procesos.

Sin embargo, con la información manejada por el EBWM, no se pueden aprender las causas de las ejecuciones de acciones en los procesos. En aplicaciones como el problema del modelado de Vías Clínicas es necesaria la recolección de los resultados de los procesos. Por ejemplo, cualquier acción que se pueda tomar en medicina, es susceptible de ser tomada a causa del resultado de una prueba. En el caso de tomar una aspirina puede venir determinado porque al tomar la temperatura se haya detectado fiebre. Este sencillo problema no puede ser solucionado por el paradigma del EBWM ya que en sus corpus no se encuentra la información relativa al resultado de la prueba *tomar temperatura*. En estos casos, el paradigma EBWM inferiría que tras haber tomado la temperatura, a veces se toma una aspirina y a veces no, pero no se recogería la razón. Esta gran limitación hace necesaria la formulación de un nuevo paradigma que permita abordar este tipo de problemas.

5.2. Aprendizaje de Flujos de Trabajo Basado en Actividades

La metodología actual de WF está fundamentalmente basada en el concepto de *evento* . Un evento es un suceso ocurrido en un instante de tiempo y que queda registrado. Un evento no tiene resultados sino que es una resultado en si mismo. Los eventos marcan hitos alcanzados por los procesos, como puede ser el principio o el fin de una acción el el conjunto del proceso. El modelo basado en eventos asume que en una secuencia cuando una acción de un proceso termina, la acción siguiente se determina por razones estadísticas. Sin embargo, en casos complejos como el de las Vías Clínicas esto no es siempre así. En muchos casos, en una secuencia las acciones siguientes vienen determinadas del modo en como acabaron las acciones anteriores. Como ya se ha dicho, el resultado de las acciones no se encuentra registrado en los WLogs basados en eventos. Por esto, la necesidad de un nuevo modelo de aprendizaje de WF capaz de cubrir las necesidades de los problemas complejos como es el caso de las Vías Clínicas, nos lleva a incidir en la necesidad de enriquecer los WLogs existentes para permitir a los algoritmos de inferencia inferir las reglas de paso entre las acciones. Este enriquecimiento nos lleva a pasar de un modelo basado en eventos a un modelo basado en el concepto de *actividad*.

Una actividad es una acción que empieza en un tiempo determinado, termina en un tiempo distinto y superior y que genera un resultado asociado. En contraposición a un evento, la actividad tiene una duración y un resultado generado.

En esta linea, se propone el paradigma del Aprendizaje de Flujos de Trabajo Basada en Actividades (ABWM, del inglés *Activity-Based Workflow Mining* [17]. Los WF modelados con actividades en lugar de eventos permitirán representar el paso de una acción a otra dependiendo del resultado de la acción ejecutada.

El ABWM toma en consideración los resultados de las ejecuciones de las actividades ejecutadas en los sistemas de WF. Esto nos va a permitir conocer el proceso con una mayor información. Esta metodología aprende WFs que permiten un guiado más efectivo basado, no solo en la ordenación de la ejecución de las acciones, sino también en las causas de esa ordenación, permitiendo una mayor comprensión del sistema.

Para poder aplicar ABWM a un problema es necesario recoger la información relativa a las actividades ejecutadas en instancias de WF pasadas. Los datos mínimos necesarios para la aplicación de este paradigma son:

- El evento de **inicio de la actividad**, que es la fecha y hora de inicio de la actividad, el nombre de la actividad y la instancia de WF asociada.

- El evento de **fin de la actividad**, que es la fecha y hora de fin de la actividad, el nombre de la actividad y la instancia de WF asociada.

- El **resultado de la actividad**, que es un valor continuo o discreto que representa el resultado que se alcanzó tras la ejecución de la actividad

5.3. Aplicaciones del Aprendizaje de Flujos de Trabajo Basada en Actividades

Los modelos de aprendizaje de WF basados en actividades por sus características ofrecen un nuevo campo de aplicación al marco general del aplicaciones del WM. Aplicaciones que por su estructura interna no encontraban demasiada utilidad en la utilización de los modelos de WM basado en eventos ahora pueden encontrar una clara ventaja en la metodología basada en actividades.

En esta sección se van a enumerar a modo de ejemplo algunos de los campos donde puede resultar especialmente atractiva la utilización de modelos basados en Actividades. Estos campos son:

- Aprendizaje de Vías Clínicas.- Las Vías Clínicas de las que ya se ha hablado anteriormente son claramente beneficiadas por el uso de técnicas de ABWM frente al EBWM. Esto es porque las acciones ejecutadas en los procesos de Vías Clínicas están usualmente basadas en los resultados de las pruebas realizadas.

- Planes de Frecuencia Alimentaria en Nutrición y Dietética.- En el campo de la nutrición, la variabilidad genética de tipos de personas y como los productos alimentarios afectan a su salud es un tema ampliamente estudiado. La dietética actualmente se basa en el estudio individualizado de los efectos de la alimentación en las personas. La información de la ingesta alimentaria se recoge mediante los cuestionarios de frecuencia alimentaria. Los cuestionarios de frecuencia alimentaria [111] son herramientas que permiten recoger la frecuencia de consumo de porciones de alimentos durante un periodo de tiempo determinado. Esta información, unida a la información de la salud general, como el peso, el Índice de Masa Corporal (IMC) o el porcentaje de grasa corporal, puede servir para calcular los efectos que la ingesta de alimentos en los individuos. Utilizando algoritmos de WM se podrían buscar buenas practicas la creación de recomendaciones alimentarias por grupos de personas que funcionaron en casos anteriores. Esto se puede hacer seleccionando las personas que han tenido efectos similares ante las mismas ingestas alimentarias.

  El EBWM no es capaz de representar la causalidad requerida en estos modelos. Por ejemplo, el resultado de medir el índice de masa corporal es vital para saber que tipo de dieta ha que seguir. El modelo basado en eventos no tendría en cuenta el resultado de la acción, por no que no es útil para este problema.

  En este caso, el ABWM si que es capaz de inferir este tipo de resultados en los WF de la ingesta de alimentos y sus efectos en grupos de personas, (IMC, peso, volumen, equilibrio hídrico...), por lo que puede ser una buena herramienta para el dietista.

- Sistemas de Modelado de Comportamiento Humano.- Existen muchos campos donde el modelado del comportamiento humano es estudiado. Por ejemplo, en el campo de la medicina, se espera que permita ayudar el diseño de mejores interfaces de usuario adaptables, mejorar la efectividad de los tratamientos, detectar situaciones de riesgo personalizadas, entre muchas otras utilidades. En el campo de la sociología se estudia para entender las decisiones humanas. Incluso en el campo de la industria de los videojuegos se estudia para mejorar el comportamiento de los personajes

virtuales en los juegos de ordenador. En general, estos sistemas pretenden encontrar los patrones que permitan conocer más sobre el hombre y su comportamiento en circunstancias tanto usuales como especiales.

El comportamiento del hombre puede ser modelado utilizando sistemas basados en WF. Sin embargo, el principal problema inherente al modelado del comportamiento humano es su excepcional complejidad y variabilidad. Esto hace que el diseño manual sea especialmente complejo. La utilización de modelos de WM serían de gran utilidad para ayudar a los diseñadores a crear los modelos.

Si el comportamiento humano fuese visto como una serie de eventos, las reacciones serían consideradas de un modo aleatoriamente independientemente del resultado otras acciones. Por ejemplo, en este caso, ante un sensor de toma de temperatura se consideraría que el humano aleatoriamente enciende o apaga el aire acondicionado. El modelo basado en actividades notaría que en los casos en que la temperatura es baja, el humano enciende el aire acondicionado en modo calor, y en caso que la temperatura sea alta, el humano enciende el aire acondicionado en modo frío.

5.4. Implantación de Sistemas de Aprendizaje de Flujos de Trabajo Basado en Actividades para el diseño de Vías Clínicas

Para poder desarrollar un sistema de ABWM que permita el apoyo al diseño de Vías Clínicas es necesario tener en cuenta diversos factores requeridos para facilitar una correcta implantación. Estos factores son los siguientes:

- Los modelos de representación de WF que requieren las Vías Clínicas requieren una muy alta expresividad y gran legibilidad. Además, ya que se está tratando de poner en marcha un modelo de aprendizaje inductivo, es necesario que el modelo de representación tenga una complejidad gramatical que facilite el aprendizaje. Por otro lado, el modelo seleccionado deb ser capaz de incluir en su esquema los resultados de las actividades como reglas de cambio entre las acciones de forma acorde al modelo de ABWM. En esta línea, para implantar un sistema de este tipo, hay que escoger un buen sistema de Representación que cumpla con estos requisitos y lo haga de la manera más eficiente posible.

- La cantidad de información existente el los WLogs de los sistemas de interpretación actuales es insuficiente para hacer funcionar un modelo basado en actividades. De este modo es necesario seleccionar un nuevo modelo de interpretación de WF capaz de almacenar esta información en sus WLogs. Por otro lado, sería muy útil que los modelos de interpretación incorporasen sistemas de simulación para probar los sistemas antes de implantarlos y para poder generar corpus de prueba para los algoritmos de aprendizaje.

- Para poder inferir WF según la metodología basada en actividades es necesario crear nuevos algoritmos que utilicen la nueva información almacenada en los sistemas. Además es necesario crear un marco de evaluación de algoritmos de WM para poder evaluar la eficacia de estos.

Todas estos factores serán abordados en los siguientes capítulos de esta memoria, realizando propuestas con el fin de proveer de las herramientas, algoritmos y metodologías necesarias para poner en marcha un sistema de ABWM para el apoyo al diseño de Vías Clínicas.

Capítulo 6

Automata Paralelo Temporizado

Los modelos actuales de representación se han orientado a resolver los problemas de diseño de WF manualmente donde la expresividad y legibilidad son conceptos cruciales. Sin embargo, las tendencias actuales se dirigen a proponer modelos de apoyo al diseño mediante la inferencia de WF que ayuden a los expertos a estandarizar los procesos. Sin embargo, en este ámbito, la falta de formalidad o la excesiva complejidad son algunos de los problemas que aquejan estos modelos.

La utilización de modelos de representación basados en la teoría de lenguajes formales nos va a permitir utilizar técnicas formales clásicas de aprendizaje inductivo aplicadas al problema del WM.

En este capítulo se propone una nueva herramienta de representación de WF basada en la teoría de autómatas y lenguajes formales que cuenta con una buena expresividad y legibilidad todo esto manteniendo la una baja complejidad gramatical. Esta herramienta pretende ser una alternativa a los modelos actuales de representación de WF ya que no solo tiene gran capacidad de representación, sino que es fácilmente ejecutable y permite el aprendizaje con técnicas clásicas de aprendizaje inductivo.

6.1. Introducción

Los sistemas de Representación de WF actuales se han basado en la proposición de lenguajes pensados para la descripción manual de los procesos. Sin embargo, desde que surgió la idea de apoyar este diseño mediante técnicas de aprendizaje inductivo los lenguajes de representación han cobrado un mayor interés. La importancia del apoyo al diseño de WF hace necesaria una revisión de los modelos de representación para facilitar esta característica.

Según los estudios realizados en la parte de antecedentes, los modelos teóricos formales han resultado ser los mejor posicionados para ser capaces de representar Vías Clínicas. En concreto, como ya se mostró, los modelos de estados finitos cuentan con ventaja en cuanto a legibilidad ya que son mas parecidos a los lenguajes conocidos por los expertos y facilidad de interpretación ya que tienen una complejidad menor. Por otro lado, la utilización de este tipo de modelos de representación podrían facilitar el aprendizaje debido a los innumerables marcos teóricos y prácticos existentes de este tipo de formalismos. Sin embargo, en la literatura no existen modelos de estados finitos de complejidad regular que representen paralelismo y capaces de representar el tiempo, características indispensables

para la representación de Vías Clínicas.

En esta línea es necesaria la creación de un modelo basado en autómatas finitos que que cumpla con las condiciones de representación que requieren los modelos de Vías Clínicas. De este modo, las características básicas esperadas de este formalismo son:

- **Expresividad**.- El lenguaje de representación seleccionado, ha de tener una gran expresividad ya que los sistemas de Vías Clínicas son potencialmente muy complejos. Además, este modelo ha de ser capaz de representar el tiempo, ya que en los modelos de Vías Clínicas, el tiempo es una variable muy usada, por ejemplo para esperar la reacción de los medicamentos.

- **Legibilidad**.- En cuanto a la legibilidad, el modelo ha de ser lo más legible posible, ya que los WF van a ser definidos por los propios facultativos.

- **Basado en actividades**.- Para poder utilizar el marco del ABWM, el formalismo presentado ha de ser capaz de representar las actividades, esto es, las transiciones entre acciones han de estar etiquetadas por los resultados de estas.

- **Complejidad Gramatical**.- Para poder inferir y ejecutar lo más fácilmente posible los modelos estos han de ser lo más simples posibles en cuanto a complejidad gramatical. La utilización de modelos basados en gramáticas simples facilitan la ejecución de los sistemas. Además, la utilización de sistemas de complejidad gramatical regular puede facilitar la inferencia.

En este capítulo se va proponer un modelo de representación de WF con una gran expresividad, legibilidad, basado en actividades y con una complejidad gramatical baja para facilitar el aprendizaje.

6.2. Autómata Paralelo Temporizado

Según las necesidades presentadas en el capítulo anterior, la propuesta planteada consiste en la definición de un modelo formal de representación de WF basado en los modelos de estados finitos que permita representar tanto patrones paralelos como representar el tiempo. Por otro lado se busca que la complejidad gramatical del modelo presentado se equivalente a la de los autómatas finitos deterministas, es decir, que el modelo sea regular. Para efectuar la representación del tiempo se va a optar por representación del tiempo discreta. Para ello se realiza la siguiente definición:

Definición 6.1 *Un Reloj C es una máquina tal que pasado un intervalo de tiempo i genera un símbolo* $t \in T$

El reloj es una máquina determinista que es capaz de generar etiquetas que describen el tiempo transcurrido. Ejemplos son:

- C^{10} genera el símbolo t^{10} 10 segundos después
- C^{0} genera el símbolo t^{0} instantáneamente
- C^{∞} nunca generará ningún símbolo □

Las etiquetas definidas son capaces de describir el tiempo de una forma discreta. Esto va a permitir que los formalismos que incorporen el concepto de Reloj no vean aumentada su complejidad más allá de los lenguajes regulares a causa del modelado del tiempo. En los sistemas de Vías Clínicas el modelado de tiempo discreto es suficiente ya que la descripción del tiempo en este tipo de sistema está basada en la conclusión de hitos temporales para la selección de las siguientes actividades. Por ejemplo esperar la reacción de un medicamento, tiempo de espera máximo para realizar una operación, realizar una tarea periódicamente, etc.

En esta línea, se propone el Autómata Paralelo Temporizado (TPA, del inglés *Timed Parallel Automaton*) [17] como una propone alternativa para la representación de WF. El TPA es una generalización de la definición del PFA [96]. Esta generalización se realiza en dos dimensiones a) representando acciones y relaciones entre actividades y b) modelando el tiempo

Por un lado, las acciones en realización se representan incorporadas en el concepto de *nodo*. Las relaciones, que vienen marcadas por los resultados de las acciones, vienen representadas mediante el etiquetado de las transiciones entre nodos. Por otro lado, el modelado del tiempo se realiza mediante el concepto de reloj.

En esta línea, se realiza la siguiente definición formal:

Definición 6.2 *Un Automata Finito Paralelo Temporizado Regular es un tupla* $A = \{C, P, N, Q, T, \Phi, \Sigma, \gamma, \delta, q_0, F\}$ *donde:*

C *es un conjunto finito de relojes,*

P *es un conjunto finito de procesos,*

$N \subseteq P \times C^+$ *es un conjunto finito de nodos,*

$Q \subseteq N^+$ *es un conjunto finito de estados,*

T *es un conjunto finito de etiquetas de tiempo,*

Φ *es un conjunto de Indicadores,*

$\Sigma \subseteq T^* \cup \Phi^+ \cup T^* \times \Phi^+ \cup \{\lambda\}$ *es el alfabeto finito de entrada,*

$\gamma : N^+ \times \Sigma \rightarrow N^+$ *es la función de transición de los Nodos,*

$\delta : Q \times \Sigma \rightarrow Q$ *es la función de transición de los estados,*

$q_0 \in Q$ *es el estado inicial,*

$F \subseteq Q$ *es el conjunto de estados finales.*

El concepto de *Nodo* en un TPA incluye un proceso p que representa una acción a automatizar y un conjunto de relojes que generaran su conjunto de etiquetas temporales, pertenecientes al conjunto T, cuando haya pasado el tiempo marcado desde que el nodo fue accedido. Las transiciones entre Nodos son una función dependiente de los Indicadores pertenecientes a Φ, que son los resultados de las acciones incluidas en los nodos, y las etiquetas temporales generadas por los relojes. Las transiciones de los nodos vienen regladas

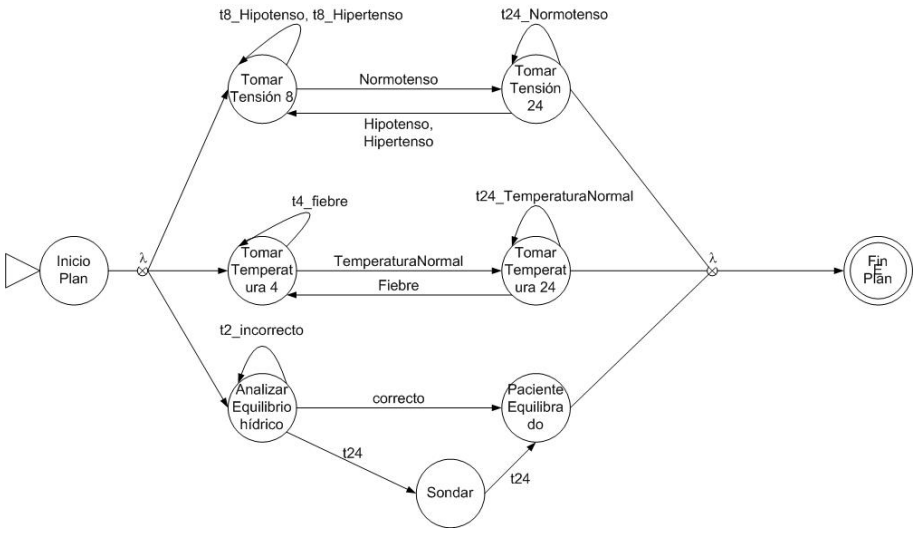

Figura 6.1: Ejemplo de modelado de una Vía Clínica usando un TPA

según la función γ. Esta función define la transición desde un grupo de nodos a otro grupo de nodos.

El concepto de *Estado* en un TPA representa la agrupación de varios nodos. La transición entre estados está definida según marca la función δ. De forma diferente a la función γ, la función δ define la transición entre dos estados dependiendo de los indicadores y los relojes asociados a sus nodos.

En la Figura 6.1, se muestra un ejemplo de uso de un TPA en un proceso de Vias Clínicas. En este proceso se intenta controlar la temperatura, la tensión y el equilibrio hídrico de un paciente ingresado. Al iniciar el plan se ejecutan paralelamente las actividades tomar temperatura, tomar la tensión, y analizar el equilibrio hídrico. En el caso de la toma de tensión, según la Figura, la tensión se ha de tomar cada ocho horas para los hipertensos e hipotensos, y veinticuatro horas para los normotensos. De manera similar, en el caso de la toma de temperatura, si hay fiebre, la acción se realiza cada cuatro horas, y si no hay fiebre se toma cada veinticuatro horas. En ambos casos, en caso de recaídas se vuelve al estado de realizar medidas frecuentemente. Por último en el caso del equilibrio hídrico, se analiza el equilibrio hídrico cada dos horas hasta que sea correcto, caso en el que el paciente estaría equilibrado. Si pasan veinticuatro horas y sigue sin corregirse se sonda al paciente y veinticuatro horas más tarde pasaría automáticamente a estar equilibrado. Una representación formal del automata es presentada a continuación:

$A = \{\mathrm{C}, \mathrm{P}, \mathrm{N}, \mathrm{Q}, \mathrm{T}, \Phi, \Sigma, \gamma, \delta, q_0, \mathrm{F}\}$,

$C = \{c^{\infty}, c^2, c^4, c^8, c^{24}\}$,

$P = \{$InicioPlan, TomarTension8, TomarTension24, TomarTemperatura4, TomarTemperatura24, AnalizarEquilibrioHidrico, Sondar, PacienteEquilibrado, FinPlan$\}$,

$N = \{n_0$:[InicioPlan, c^{∞}],
n_1:[TomarTension8,c^8],

n_2:[TomarTension24, c^{24}],
n_3:[TomarTemperatura4, c^4],
n_4:[TomarTemperatura24, c^{24}],
n_5:[AnalizarEquilibrioHidrico, c^2c^{24}],
n_6:[Sondar, c^{24}],
n_7:[PacienteEquilibrado, c^∞],
n_8:[FinPlan, c^∞]}

Q = {$q_0 : n_0$, $q_{135} : n_1n_3n_5$, $q_{235} : n_2n_3n_5$, $q_{145} : n_1n_4n_5$, $q_{136} : n_1n_3n_6$, $q_{137} : n_1n_3n_7$, $q_{245} : n_2n_4n_5$, $q_{236} : n_2n_3n_6$, $q_{237} : n_2n_3n_7$, $q_{146} : n_1n_4n_6$, $q_{147} : n_1n_4n_7$, $q_{246} : n_2n_4n_6$, $q_{247} : n_2n_4n_7$,$q_8 : n_8$},

T ={t^2, t^4,t^8,t^{24}},

Φ = {Hipotenso, Hipertenso, Normotenso, Fiebre, TemperaturaNormal, Correcto, Incorrecto},

Σ = {Hipotenso, Hipertenso, Normotenso, Fiebre, TemperaturaNormal, Correcto, Incorrecto, t^2, t^4, t^8, t^{24}, t^8Hipotenso, t^8Hipertenso, t^4Fiebre, t^2Incorrecto, t^{24}Normotenso, t^{24}TemperaturaNormal, λ},

γ : {[n_0, λ, $n_1n_3n_5$], [n_1, t^8*Hipotenso*, n_1], [n_1, t^8*Hipertenso*, n_1], [n_1, Normotenso, n_2], [n_2, t^{24}*Normotenso*, n_2], [n_2, Hipertenso, n_1], [n_2, Hipotenso, n_1], [n_3, t^4Fiebre, n_3], [n_3, TemperaturaNormal, n_4], ...},

δ : {[q_0, λ, q_{135}], [q_{135}, t^8Hipotenso, q_{135}] [q_{135}, t^8Hipertenso, q_{135}], [q_{135}, t^4Fiebre, q_{135}], [q_{135} ,t^2Incorrecto, q_{135}], [q_{135}, Normotenso, q_{235}], [q_{135}, TemperaturaNormal, q_{145}], [q_{135}, t^{24}, q_{136}], [q_{135}, Correcto, q_{137}], [q_{235}, t^{24}Normotenso, q_{235}], [q_{235}, t^4Fiebre, q_{235}], ... }

q_0 = {q_0},

F = {q_8}

En esta descripción forma el conjunto de relojes C contienen los modelos de reloj que van a ser necesarios para la especificación del problema, en este caso, van a ser necesarios relojes de 2, 4, 8 y 24 horas mientras que el reloj c^∞ se encontraran en los nodos que no dependan del tiempo para su transición. Las etiquetas generadas por estos relojes se encuentran en el conjunto T.

Los procesos a ejecutar se encuentran en el conjunto P. El conjunto de nodos asigna a cada proceso un conjunto finito de relojes que forma cada nodo. Por ejemplo, el Nodo n_5 que representa la acción de *AnalizarEquilibrioHidrico* tiene asociados dos relojes diferentes(c^2 y c^{24}) debido a que se ha de realizar la acción cada 2 horas pero en el caso de que este proceso durase más de 24 horas se ha de sondar al paciente.

El conjunto de estados Q marca todas las posibles combinaciones de nodos que pueden ejecutarse a la vez en un momento dado. El estado q_{135} describe que los nodos n_1, n_3 y n_5 están ejecutándose a la vez. De esta manera se pueden expresar que las acciones *TomarTension*, *TomarTemperatura* y *AnalizarEquilibrioHidrico* se ejecutan en paralelo. El conjunto Φ muestra el conjunto de posibles resultados que pueden derivarse de los

procesos en ejecución. En el caso de *TomarTension* los resultados posibles son *Hipertenso*, *Normotenso* e *Hipotenso*.

La función de transición γ marca las transiciones entre los nodos. Por ejemplo [n_1, Normotenso, n_2] significa que desde el nodo n_1 se transita al nodo n_2 cuando el resultado del proceso asociado al nodo n_1 (*TomarTension*) da como resultado Normotenso. Otro ejemplo de transición utilizando relojes es n_1, t^8*Hipertenso*, n_1. En este ejemplo, se transita del nodo n_1 a el mismo, cuando el resultado de *TomarTension* sea *Hipertenso* y el reloj c^8 se haya cumplido. De este modo representaremos que mientras el resultado de la acción de tomar la tensión sea hipertenso, se ha de repetir la acción cada 8 horas. Por ultimo un ejemplo de acción que exprese una separación paralela es [n_0, λ, $n_1n_3n_5$]. Según este ejemplo, se pasa del nodo inicial, los nodos n_1, n_3 y n_5.

Por su parte la función δ marca las transiciones entre los estados. De este modo, [q_{135}, Normotenso, q_{235}] significa que se transita desde el conjunto de nodos q_{135} al conjunto de nodos q_{235} cuando el proceso *TomarTension* tiene como resultado Normotenso. Nótese que esta transición está relacionada con la transición a nivel de nodo [n_1, Normotenso, n_2]. Esto, las transiciones a nivel de nodo solamente se ejecutan de forma colaborativa con las transiciones a nivel de estado

Para terminar, se especifica el estado inicial, y los estados finales. □

6.2.1. Definiciones Formales del TPA

En este apartado se van a definir formalmente diversas estructuras que harán mas sencilla la explicación de las características del TPA en su aplicación. Las definiciones realizadas se refieren a estructuras básicas que tienen utilidad en la interpretación y aprendizaje de WF. Las definiciones son las siguientes:

Definición 6.3 *Dado un TPA* $A = \{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$

$Transicion(i) = \{(q_i, t, q_f)\ donde\ q_i, q_f \in Q \wedge t \in \Sigma | \exists \gamma_n = (q_i, t, q_f)\}$

$Token(Transicion(i)) = \{t \in \Sigma | t \in Transicion(i)\}$

Una transición no es mas que la descripción del paso de un estado a otro basándose en la función de transición de los estados producida. Como veremos más adelante cuando analicemos el comportamiento del modelo, aunque la transición de un TPA se base en la transición de los estados, existe una correlación entre función de transición de los estados y la función de transición de los nodos. Por otro lado se denomina *token* al conjunto de indicadores y etiquetas de reloj necesarias para la derivación de una transición □

Definición 6.4 *Dado un TPA A y una Transición* $T \in Transiciones(A)$

$Dominio(T) = \{\{n_0, .., n_i\} \in q_i | T = (q_i, a, q_f)\}$

$Rango(T) = \{\{n_0, .., n_i\} \in q_f | T = (q_i, a, q_f)\}$

Se denomina *Dominio de una Transición* o nodos origen de una transición de un TPA al conjunto de nodos que pertenecen al estado del que parte una transición. Por su parte, se denomina *Rango de una Transición* o nodos destino de una transición de un TPA al conjunto de nodos que pertenecen al estado en el que deriva una transición. □

Definición 6.5 *Dado un TPA A, y un Nodo $n \in N$*

$Dominio(n) = \{\{n_0, .., n_i\} \in \cup q_i | \forall t_A = (q_i, a, q_f) \in Transicion(A) \wedge n \in q_f\}$

$Rango(n) = \{n_0, .., n_i\} \in \cup q_f | \forall t_A = (q_i, a, q_f) \in Transicion(A) \wedge n \in q_i\}$

De forma análoga a las transiciones, se denomina *Dominio de un nodo* de un TPA al conjunto de nodos desde los que se puede transitar a otro nodo utilizando una única transición Por otro lado, se denomina *Rango de un Nodo* de un TPA al conjunto de nodos a los que se puede transitar desde un nodo utilizando una única transición. □

Definición 6.6 *Dado un TPA A = {C, P, N, Q, T, Φ, Σ, γ, q_0, F} y una traza de A $t_A \in$ Transiciones(A);*

$EstadoActivo(t_A) = \{q_i \in Q \wedge \exists(q_i, a, qi + 1, t_0, \emptyset) \in Transiciones(t_A)\}$

$NodoActivo(t_A) = \{n \in EstadoActivo(t_A)\}$

Cuando un TPA se encuentra funcionando como un WF en ejecución y pasa de un estado a otro, se van activando nodos y estados conforme se van alcanzando. Un estado de un TPA de dice que es un *Estado Activo* cuando el flujo del WF en ejecución actualmente ha alcanzado dicho estado. Un TPA solo tiene un estado activo al mismo tiempo. Análogamente, los nodos de los estados activos se denominan *Nodos Activos.* □

Definición 6.7 *Dado $q \in Q$ y a un token de entrada tal que $a \in \Sigma$,*

$$\delta(q, a) = \{(q - m) \cup n | n = \gamma(m, a) \, para \, m \subseteq q\}$$

La definición 6.7 se refiere al comportamiento del TPA. En esta función se muestra la forma en la que el TPA se comporta cuando se produce una transición. Aunque como hemos visto antes, las reglas de la transición son siempre de acuerdo a la función de transición de estados (δ) esta función esta correlacionada con la función de transición de nodos (γ). La relación está descrita en esta definición. Intuitivamente, cada estado activo gestiona un conjunto de nodos y sus transiciones, cuando uno o varios de los nodos activos junto con la palabra de entrada cumple una de las transiciones válidas según la función γ, estos nodos origen se desactivan y se activan los nodos resultado de la función de transición de nodos. Como resultado tenemos el resultado de la función γ unido con el resto de los nodos origen del estado que no han participado en la transición de γ. En el ejemplo de la Figura 6.1, la transición [q_{135}, *Normotenso*, q_{235}] perteneciente a δ y la transición [n_1, *Normotenso*, n_2] perteneciente a γ definen el comportamiento del TPA cuando se están ejecutando los procesos *TomarTension*(n_1), *TomarTemperatura*(n_3) y *AnalizarEquilibrioHidrico*(n_5) y se produce como resultado del proceso *TomarTensión* el indicador *Normotenso*. En este caso, el nodo n_1 que es origen en la transición de γ se desactiva y el nodo n_2 que es destino es esta transición se activa. De este modo, los nodos activos después de la derivación son el nodo n_2, n_3 y n_5 que pertenecen al estado q_{235} que es justamente el resultado esperado según la función δ. □

Definición 6.8 *Dado un estado $q \in Q$ y $a\alpha$ una palabra de entrada $\in \Sigma^*$*

$$\delta(q, a\alpha) = \{\delta(p, \alpha) | p = \delta(q, a)\}$$

En esta definición se extiende el concepto de δ a la definición de multiples derivaciones. Según esta definición una palabra, formada por varios tokens de entrada puede ser subdividida para ir aplicando sucesivamente las derivaciones conforme a las reglas de comportamiento de la definición 6.7. □

Definición 6.9 *El lenguaje aceptado por un TPA (M) se puede definir como:*

$$L(M) = \{w | \delta(q_0, w) \in F\}$$

Basándose en la definición 6.8 se puede describir el lenguaje aceptado por un TPA al conjunto de palabras de que desde en estado inicial derivan en un estado final.

□

6.2.2. Características formales del TPA

En este apartado se van describir y demostrar teoremas que nos permitan describir las características del TPA en cuanto a su complejidad gramatical y los marcos formales a los que pertenece y así poder conocer las herramientas teórico-prácticas a las que es posible acogerse. Las características formales del TPA son:

Teorema 6.1 *La clase de los lenguajes aceptados por los automatas paralelos temporizados(TPA) es equivalentes a la clase de los lenguajes aceptados por los Automatas Finitos Paralelos(PFA)*

Prueba 6.1 *$PFA \subseteq TPA$: Se puede construir un TPA a partir de un PFA. Dado un PFA tal que PFA=$\{N_P, Q_P, \Sigma_P, \gamma_P, \delta_P, q_{0_P}, F_P\}$ existe un TPA tal que $TPA = \{C, P, N, Q, T, \Phi, \Sigma, \gamma, \delta, q_0, F\}$ donde:*

$C = \emptyset$, $P = \emptyset$,
$N = N_P$,
$Q = Q_P$,
$T = \emptyset$,
$\Phi = \Sigma_P$,
$\Sigma = \Sigma_P$,
$\gamma = \gamma_P$,
$\delta = \delta_P$,
$q_0 = q_{0_P}$, $F = F_P$

$TPA \subseteq PFA$: Del mismo modo un TPA puede ser genéricamente formado a partir de un PFA, Dado un TPA =$\{C_t, P_t, N_t, Q_t, T_t, \Phi_t, \Sigma_t, \gamma_t, \delta_t, q_{0_t}, F_t\}$ existe un PFA tal que PFA=$\{N, Q, \Sigma, \gamma, \delta, q_0, F\}$ donde:

$N = N_t$,
$Q = Q_t$,
$\Sigma = \Sigma_t$,

$\gamma = \gamma_t$,
$\delta = \delta_t$,
$q_0 = q_{0_t}$, $F = F_t$

□

Este teorema muestra las equivalencias en cuanto a complejidad gramatical propias del TPA con el PFA. Según este teorema, las clases de los lenguajes del TPA y PFA son equivalentes. Por esto, podemos concluir que se ha podido incluir la información temporal en el modelo sin aumentar complejidad gramatical. Como corolarios de este teorema tenemos dos consecuencias directas de este. Estos dos corolarios están referidos a los TPA y su equivalencia con los lenguajes regulares y las Redes de Petri

Corolario 6.1 *La clase de los lenguajes definidos por los TPA son equivalentes a la clase de los lenguajes regulares*

Prueba 6.2 *En [96] se demuestra que un PFA es equivalente a un AFN. Como se demostró anteriormente la clase de lenguajes admitidos por un TPA es equivalente a la clase de lenguajes admitidos por un PFA.*

□

Corolario 6.2 *El lenguaje admitido por una Red de Petri 1-segura, es admitido por un TPA*

Prueba 6.3 *En [96] se demuestra que un PFA es equivalente a una Red de Petri 1-segura. Como se demostró anteriormente la clase de lenguajes admitidos por un TPA es equivalente a la clase de lenguajes admitidos por un PFA*

□

6.3. Expresividad de un TPA

El siguiente paso después de haber formalizado el TPA es medir su expresividad. La usual métrica utilizada para medir la expresividad según la literatura son los patrones de WF. Los patrones de WF son un conjunto de descripciones de situaciones posibles que se pueden encontrar en un proceso real. Estos patrones son utilizados para distinguir la capacidad expresiva de los sistemas de representación de WF. Cuantos más patrones de WF sea capaz de representar un modelo mas expresivo será.

Los patrones de WF miden no solo los sistemas de representación sino también los sistemas de interpretación. En el caso que nos ocupa vamos a medir únicamente la capacidad expresiva del modelo de representación presentado. Para ello, se va a estudiar la capacidad expresiva que tiene el TPA para representar los patrones de Flujo de Control, y dentro de ellos los específicamente relacionados con la capacidad expresiva de los sistemas de representación. En este apartado se va a describir a grandes rasgos como el TPA cumple los patrones de WF. En esta sección se van a analizar los patrones por grupos principales[1]. Los patrones analizados son los siguientes:

[1]Para una información más detallada, en el anexo A se analizan con detalle los patrones de WF y se demuestra formalmente la adecuación del TPA de los patrones orientados a los sistemas de representación.

- Patrones de Flujo Básico.- Los patrones de flujo básico son los que describen las situaciones más utilizadas en los procesos reales. Dentro de estos Patrones se encuentran los patrones de *Secuencia*, *Separación Paralela*, *Sincronización*, *Elección Exclusiva* y *Fusión Simple*. El TPA es capaz de representar correctamente todos estos patrones. La demostración formal se encuentra en la sección A.1 del anexo A.

- Patrones de Ramificación Avanzada y Sincronización.- Estos patrones describen situaciones complejas que definen ramificaciones y sincronizaciones de los flujos. Entre estos patrones están los patrones de *Multielección*, *Fusión Sincronizada*, *Multifusión* y *Discriminador*. El TPA es capaz de representar correctamente todos estos patrones. La demostración formal se encuentra en la sección A.2 del anexo A.

- Patrones Estructurales.- Estos patrones describen situaciones que son vistas como apoyo estructural al flujo de los WF. Hay dos tipos de patrones estructurales, los *Ciclos Arbitrarios* y *Terminación Implícita*. EL TPA es capaz de representar correctamente el patrón de ciclos arbitrarios. En cuanto al patrón de Terminación Implícita, este patrón se refiere a la capacidad del motor de interpretación para terminar un modelo cuando detecta que no hay ninguna acción que realizar. Por eso este patrón no es objetivo del sistema de representación. Las demostraciones formales de encuentran en la sección A.3 del anexo A.

- Patrones que involucran Multiples instancias.- Los patrones que involucran multiples instancias están pensados para evaluar la capacidad que tienen los motores de interpretación para trabajar con varias instancias en paralelo. Por esta razón estos patrones no son objetivos de esta sección.

- Patrones Basados en Estados.- Estos patrones describen situaciones en la que el sistema puede quedarse en modo de espera indefinidamente. Entre estos, se encuentran los patrones de*Elección Derivada*, *Rutina Paralela Entrelazada* e *Hito*. El TPA no tiene problemas en representar estos patrones como queda demostrado en la sección A.5 del anexo A.

- Patrones de Cancelación.- Estos patrones evalúan la capacidad de los motores de interpretación para eliminar instancias y nodos en tiempo de ejecución. Por esto, estos patrones no son objetivo del TPA.

El TPA ha demostrado tener una gran expresividad siendo capaz de representar todos los patrones orientados a la especificación de los procesos. Por lo tanto es una herramienta muy válida para la especificación de WF.

6.4. Caso de aplicación del TPA: Protocolos de Actividad de Vida

En la actualidad uno de los caballos de batalla más importantes en el mundo de la salud es la prevención. Dentro de este campo uno de los modelos más novedosos son los modelos de e-salud centrados en el ciudadano. Estos modelos tienen como objetivo la prevención, el control y tratamiento de las enfermedades de un modo integrado en la vida diaria de los pacientes.

Los Protocolos de Actividad de Vida (LAP, del inglés *Life Activity Protocols*) son un modelo dentro de este contexto. Un LAP es un conjunto de guías, recomendaciones y prescripciones para una necesidad concreta del usuario incluyendo no solo acciones para el cuidado de la salud sino acciones de prevención. Esta es la principal diferencia entre las Vias Clínicas y los LAP. Mientras que los planes de cuidados están orientados al cuidado de las enfermedades post evento, los LAP son un protocolo de cuidado del paciente antes y después del evento. De este modo los LAP pueden ser utilizados para tratar, controlar o prevenir patologías (diabetes, insuficiencia cardíaca), condiciones especiales (embarazos, ancianos), prevención de la salud en general (dejar de fumar) o cualquier otro modelo que pretenda proporcionar guías para un modo de vida saludable.

Para que estos protocolos tengan sentido más allá de la prevención pasiva o recomendaciones generales de los organismos sanitarios, esta ha de ser aplicada continuamente al paciente de un modo continuo. Debido a que no es posible un cuidado continuo del ciudadano a través de cuidadores humanos es necesaria la utilización del sistemas de e-salud. Para que estos sistemas sean capaces de entender estos modelos han de ser escrito a modo de lenguaje formal que pueda ser no ambiguo, compilable y ejecutable por un computador. Por otro lado, el lenguaje ha de ser lo suficientemente expresivo para representar todos los Patrones de WF existentes en la vida diaria de los pacientes. Además ha de ser suficientemente legible como para que los profesionales de la prevención puedan diseñar los modelos sin tener que aprender tecnologías de la programación. Estas características apuntan claramente a la utilización de WF.

El TPA se tomó como base para la definición del LAP. Este modelo se utilizó con éxito en el proyecto europeo PIPS [7], y se presentó en un capitulo del libro [44].

6.5. Conclusiones y Aplicabilidad del TPA a las Vías Clínicas

En la introducción del capítulo se analizaron las características que debía de tener un modelo de representación de WF que intentara abordar el problema de la especificación del las Vías Clínicas pensando además en la utilización de modelos de aprendizaje que pudieran ayudar en el diseño de este tipo de sistemas. Estas características son Expresividad, Legibilidad, Basado en Actividades y Baja Complejidad Gramatical.

En cuanto a la expresividad, el TPA ha demostrado ser capaz de expresar todos los patrones de WF orientados a la representación de Van der Aalst. Además es capaz de expresar el tiempo de de una forma fácilmente. Los sistemas de Vías Clínicas necesitan modelos con una gran expresividad y capacidad modelado del tiempo. Por esto el TPA cumple con las expectativas cubiertas

Por parte de la legibilidad, ya en el estado del arte los modelos de estados finitos aparecían entre los modelos, a priori mas legibles. En el proyecto PIPS [7]. Un modelo de representación basado en TPA fue el seleccionado por los expertos en salud para la definición de procesos por su legibilidad y expresividad [31]. Además, el TPA por su naturaleza tiene una talla menor que los modelos basados en Redes de Petri, lo cual facilita la legibilidad frente a estos.

En cuanto a la orientación a actividades, el TPA incorpora en su modelo las acciones y sus resultados en forma de indicadores por tanto es muy fácil crear WF orientados a

actividades donde los resultados de las acciones y el tiempo decidan el flujo de ejecución de los procesos.

En cuanto a la baja complejidad gramatical, los TPA son equivalentes a los lenguajes regulares, con lo que su complejidad es relativamente baja. Esto permite la utilización de potentes marcos formales de aprendizaje que de otro modo no hubiese sido posible.

El TPA ha demostrado ser una buena alternativa para su aplicación a los modelos de Vías Clínicas orientados al aprendizaje. El siguiente paso será la propuesta de un motor de interpretación capaz de ejecutar este tipo de modelos.

Capítulo 7

Interpretación de Autómatas Paralelos Temporizados

Poner en marcha una Vía Clínica requiere herramientas que nos permitan no solo especificar los procesos sino también apoyar la ejecución de los procesos cuando estos se están ejecutando. Guiar estos procesos dinámicamente requiere la creación de sistemas de interpretación de WF que permitan automatizar las especificaciones de los procesos.

Para realizar esto, después de definir el TPA, el siguiente paso es crear un motor de ejecución que sea capaz de ejecutar especificaciones de procesos diseñadas utilizando este modelo. Además, este motor deberá ser capaz de crear WLogs basados en actividades para que el paradigma ABWM pueda ser utilizado y permitir la simulación de procesos para probar los sistemas antes de su implantación, y poder generar corpus de aprendizaje de WF facilitando así la evaluación de los algoritmos de aprendizaje. En este capítulo se va a presentar un motor de WF capaz de funcionar bajo estas premisas.

7.1. Introducción

En el capítulo anterior se definió el TPA como herramienta de representación de WF. Al ser un TPA una descripción formal, es posible crear una aplicación software que permita la ejecución automática de los procesos descritos en él. Este tipo de herramientas, se denominan *motores de interpretación* o *motores de WF*.

Para la ejecución de Vías Clínicas utilizando el TPA como modelo de representación requiere la creación de un nuevo motor de interpretación. Este motor de interpretación podría ser usado para guiar a los profesionales de la salud según los procesos definidos en los TPA. Este guiado de las tareas facilita la estandarización de los procesos.

Un motor de interpretación para TPAs que representen Vías Clínicas requiere cumplir las siguientes características:

- **No limitar la expresividad**.- Como vimos en el apartado de antecedentes, cuando los lenguajes de especificación se llevan a ejecución, en mucha ocasiones sufren recortes para permitir la interpretación de determinadas estructuras. Dadas las altas necesidades de representación esta característica es crucial para la definición de procesos de Vías Clínicas.

- **Modificación Dinámica de instancias de ejecución**.- Los sistemas de Vías Clínicas son susceptibles de verse modificados en tiempo de ejecución. Las Vias Clínicas son estandarizaciones de cuidados para pacientes que sufran una determinada patología. Sin embargo, debido a la gran variabilidad de los paciente, los tratamientos no son siempre igualmente efectivos en todos los pacientes. Por eso, es necesario que los WF puedan ser modificados dinámicamente en ejecución en los casos que sea necesario.

- **WLogs compatibles con ABWM**.- La capacidad de representar modelos basados en actividades y su complejidad gramatical regular, da a los modelos basados en TPA la posibilidad de beneficiarse de las técnicas de ABWM para la mejora constante de las especificaciones. Para esto es necesario hacer una recogida de los datos ocurridos durante la ejecución de las Vías Clínicas. Estos datos serán usados por los algoritmos de aprendizaje para que puedan presentar WFs a los expertos donde se especifique la ejecución real de los procesos, y puedan refinar las especificaciones. Los sistemas actuales de interpretación de WF recogen datos sobre los eventos ocurridos, pero no almacenan en sus WLogs toda la información necesaria para los sistemas de ABWM. Es necesario por tanto que el modelo de interpretación de WF, almacene datos referentes a los resultados de las acciones para aprovecharse de los modelos de ABWM.

- **Simulación de Procesos**.- Dotando a los motores de interpretación de WF de capacidades de simulación se consigue, por un lado, permitir una forma estándar de probar los procesos, facilitando la detección de errores antes de su implantación. Por otro lado, de esta forma se pueden crear corpus de aprendizaje que pueden ser usados por los algoritmos de WM para probar su capacidad inductiva.

En este capítulo se presenta un motor implementado para interpretar y simular TPAs de una forma eficiente y que cree WLogs con la información necesaria para que puedan ser utilizados por el paradigma del ABWM.

7.2. Modelo de Interpretación del TPA

El TPA está basado en la especificación de transiciones entre estados y nodos para el modelado del comportamiento de los procesos en un determinado entorno. Intuitivamente, la información del estado ofrece información sobre el flujo que ha de seguir en cuanto a los actividades que se encuentran activas en un determinado momento. Por otro lado la información de los nodos ofrece una visión sobre la activación o desactivación de nodos en el momento de una transición.

En la Figura 7.1 se puede observar como se producen las transiciones en un TPA. En esta figura se muestra con un ejemplo el modelo de derivación descrito en la Definición 6.7 en el capítulo donde se describe el TPA. Según esta definición, cada derivación de la función δ va a depender del estado actual, del token de entrada y de la función γ.

El ejemplo de la figura muestra una sincronización de dos secuencias paralelas. Por un lado, se ejecutan secuencialmente los nodos 1 y 3 y por otro se ejecutan los nodos 2 y 4. Estas dos secuencias se sincronizan en el nodo 5. Desde el punto de vista del estado, en la figura existen cuatro estados diferentes: el estado A que contiene los nodos 1 y 2; el

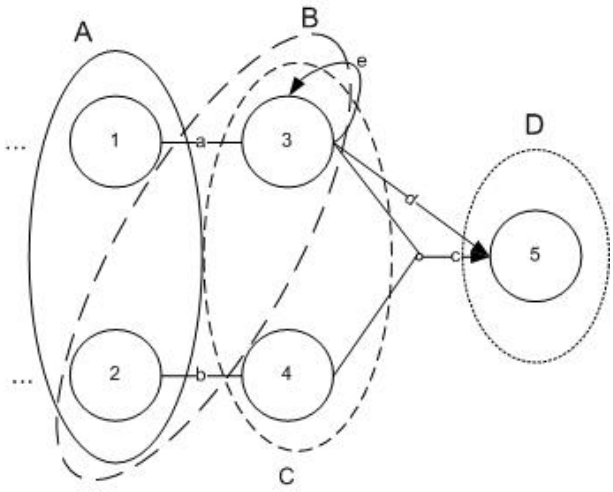

Figura 7.1: Descripción gráfica del comportamiento de un TPA en ejecución

estado B que contiene los nodos 3 y 2; el estado C que contiene los nodos 3 y 4 y el estado D donde se sincronizan las dos secuencias, y contiene el nodo 5. Suponiendo el estado A activo, al llegar un token a según la función δ pasaremos al estado B. En este mismo caso, según la función γ informa que hemos de desactivar el nodo 1 y activar el 3. El motor de interpretación debe interpretar esto como parar la ejecución del proceso embebido en el nodo 1 e iniciar el proceso del nodo 3. Sin embargo, el proceso incluido en el nodo 2 no se ve afectado. En el caso de que se pase del estado B de nuevo al estado B utilizando el ciclo del nodo 3 al nodo 3 con el token e. En nodo 3 se reiniciaría, pero el nodo 2 no se vería afectado.

De esta forma el estado mantiene un registro de los nodos activos en un determinado momento de la interpretación del TPA, y la función de transición de los nodos define que nodos han de reiniciarse impidiendo que la ejecución de unas secuencias afecte sobre la ejecución de otras.

Esta información es susceptible de ser tratada en tiempo de compilación para asociar la información de ejecución de los procesos a las transiciones de modo que se pueda construir un autómata de Mealy para ejecutar un TPA. Según este modelo, cada estado del autómata de Mealy se corresponde con un estado del TPA, y las transiciones marcan que nodos son los siguientemente activados según la función γ. Esto supone que simplemente adoptando el modelo de TPA para la representación de WF motores existentes del mercado con capacidad para ejecutar modelos basados en autómatas podrían ejecutar estos modelos. Esto ha sido probado dentro del proyecto PIPS [7] donde se ha utilizado un motor de jBPM para ejecutar TPAs [31].

7.3. TPAEngine

TPAEngine es una herramienta software que permite la ejecución y simulación de TPAs. TPAEngine está programado en lenguaje CSharp para la plataforma .NET [81]

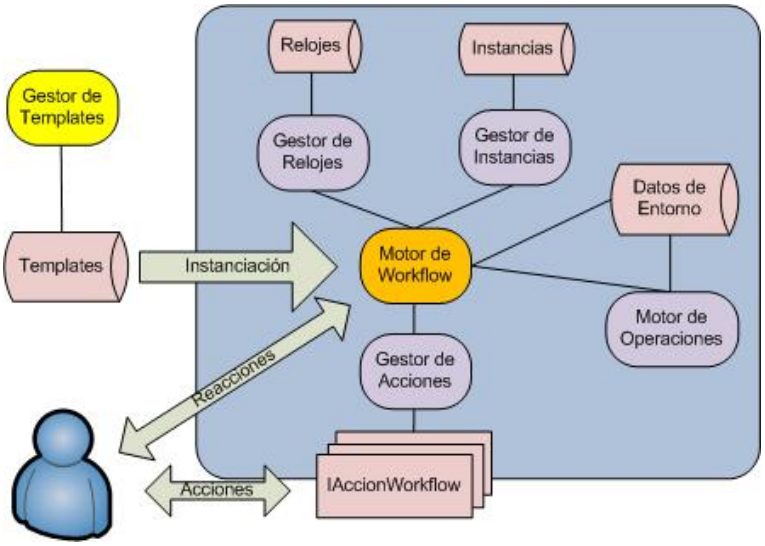

Figura 7.2: Arquitectura del TPA Engine

de Microsoft. TPAEngine permite la ejecución ordenada de actividades que se refieren a funciones nativas de CSharp. De modo que es posible construir WF que ejecuten ordenadamente métodos que realizan funciones tanto automáticamente como a través de actores humanos.

7.3.1. Arquitectura

En la Figura 7.2 se presenta la arquitectura del TPAEngine. El TPAEngine está compuesto por:

- *Gestor de Relojes.*- El Gestor de Relojes es el modulo que controla los relojes instanciados por los nodos alcanzados y comunica al sistema de la finalización de estos. El Gestor de Relojes contiene un repositorio donde los relojes van siendo almacenados conforme van siendo instanciados, y se eliminan conforme se van cumpliendo.

- *Gestor de Instancias.*- El Gestor de Instancias es el módulo que contiene las instancias de WF en ejecución. Este gestor almacena y mantiene las instancias insertadas en el motor durante todo el tiempo de vida de estas. Dado que estas instancias se almacenan copiando la información de la especificación del WF en la propia instancia es posible modificar la instancia dinámicamente sin que afecte a la especificación original.

- *Gestor de Acciones.*- El Gestor de Acciones en el módulo que invoca a las actividades realizadas en los nodos. Las acciones deben cumplir la interfaz IAccionWorkflow para poder ser ejecutadas. Estas acciones con invocadas cuando los nodos correspondientes son alcanzados. Las acciones pueden ser acciones directas sobre el sistema (como por ejemplo, programar acciones como tomar la temperatura para un momento dado o actuar sobre en el entorno en un modelo de inteligencia ambiental) o pueden ser

mensajes enviados a usuarios para que tomen decisiones (como por ejemplo informar sobre los caminos a seguir en un momento dado a un médico en una vía clínica).

- *Datos del Entorno*.- El Gestor de Datos del Entorno es un repositorio de datos donde se almacenan todos los datos referidos a las instancias que se están ejecutando en el sistema. Datos como el estado de los relojes o los resultados de las acciones son almacenados en este repositorio.

- *Motor de Operaciones*.- El Motor de Operaciones es un modulo que compila y ejecuta *just in time* las operaciones a realizar en las transiciones. El Motor de Operaciones es capaz de calcular si las transiciones se pueden o no se pueden ejecutar.

- *Motor de WF*.- El Motor de WF es el núcleo del sistema y comunica las diferentes partes para permitir la interpretación de los WF. es el encargado de instanciar los templates, iniciar los relojes, recuperar las instancias cuando son necesarias, escoger las transiciones para que sean ejecutadas, e invocar al gestor de acciones.

Para diseñar el WF, se definen los flujos en forma de estados y nodos. Las acciones de los nodos han de cumplir la interfaz IAccionWorkflow. Para ejecutar el flujo, este se instancia en el sistema, y pasa a ser controlado por el gestor de instancias. El gestor de instancias va despertando las instancias cuando se reciben nuevos datos o finalizan relojes que puedan modificar el estado actual de las instancias. Mientras estas instancias no reciben ningún dato, permanecen en estado de hibernación y no consumen recursos. Esto permite una gran escalabilidad del sistema ya que, en condiciones normales, ejecuta muy pocos flujos a la vez, manteniendo una gran cantidad de flujos en hibernación. Otros motores, que dejan la gestión del tiempo a las propias acciones, exigen que los flujos estén siempre en activo lo que afecta directamente a la escalabilidad. Otros sistemas, como los basados en WSBPEL puro, no permiten la hibernación de procesos, porque restringen que los procesos empiecen y terminen en la misma llamada no permitiendo procesos de elección diferida.

7.3.2. Modos de ejecución del TPAEngine

La interpretación de WF puede ser usada de muchas formas dependiendo del problema en cuestión. Por ello, el TPAEngine se ha diseñado pensando en la posibilidad de ser utilizado en distintos campos. De este modo, utilizando TPAEngine existen varios modos de ejecución de los procesos:

- **Automatización de procesos**.- Por un lado si no es necesaria la mediación de actores humanos, es posible al automatización completa de los procesos. En este caso, los procesos son ejecutados directamente sin tener en cuenta la necesidad de la monitorización por un agente humano. En casos de WF cerrados, como procesos que orquesten la respuesta de un sistema ante un evento se utilizan este tipo de WF. Un ejemplo de esto es la respuesta que da un sistema de incendios de un sistema de Inteligencia Ambiental ante un incendio, coordinando llamadas y guiando al usuario hacia posiciones seguras.

- **Guiado de procesos**.- Por otro lado, si es necesaria la mediación de actores humanos, cada acción es sugerida a modo de sistema experto para que el experto seleccione

los caminos a seguir. Por ejemplo, en el caso de las Vías Clínicas, los médicos continuamente están monitorizando el proceso de cuidado del paciente. En este caso, en un alto porcentaje de ocasiones (dependiente de la bondad del modelo) el paciente sigue exactamente el camino marcado por el WF, y el médico no necesita modificar la ejecución. Sin embargo, en los casos en los que el paciente se descompensa, las instancias de WF pueden ser modificadas en tiempo de ejecución para adecuar los cuidados a las distintas respuestas del paciente. Usualmente este tipo de WF están diseñados para requerir validación por parte del mediador en algunos de los pasos a seguir por el proceso, e incluso, en algunos de los casos, en cada uno de los nodos.

- **Simulación de procesos**.- Sobre este motor base se ha colocado una capa de simulación que permite crear simulaciones definiendo simplemente las acciones a realizar y sus posibles resultados con sus probabilidades de ejecución. La capa de simulación tiene dos modos de ejecución: *simulación normal* y *simulación canónica*.

  El modo de simulación normal va ejecutando una traza del flujo teniendo en cuenta las probabilidades de ejecución de las acciones realizadas. Por otro lado, el modo de simulación canónica ejecuta todos los flujos posibles del WF. Este modelo permite la generación de corpus que pueden ser utilizados para inferencia, y permite la detección de errores en los WF, ya que ejecuta todos los caminos posibles.

7.3.3. Plantillas de WF e Instancias de WF

Las plantillas de WF son las especificaciones de WF antes de ser instanciadas. Estas contienen la información del flujo a nivel generalista para posteriormente convertirse en un flujo real en ejecución cuando se instancian, convirtiéndose en Instancias de WF. Tanto las plantillas como las instancias en el TPAEngine tienen un esquema similar basado en XML.

En la Figura 7.3 se puede ver el esquema XML que se usa como formato de entrada del TPAEngine. Las instancias copian exactamente el mismo esquema de la plantilla al iniciarse el proceso. Es importante notar que el esquema de la instancia puede cambiar independientemente debido a circunstancias especiales del WF sin tener que variar la plantilla asociada. Esto es una importante mejora que permite al TPAEngine ser utilizado en dominios exigentes como son las Vías Clínicas. Las instancias de Vías Clínicas pueden variar la especificación de su flujo durante su ciclo de vida sin que sea deseable cambiar la plantilla. Muchos cambios de flujo pueden deberse a los problemas de pacientes específicos que sufren patologías más allá de de las que es capaz de tratar la Vía Clínica en cuestión. Además no es deseable que este cambio se produzca en la plantilla, porque esta plantilla perdería generalidad.

Cuando una instancia ha terminado de ejecutarse pasa a formar parte del WLogs donde se guarda el histórico de ejecuciones. Las instancias de ejecución finalizadas se denominan muestras de ejecución. Se denomina *Muestra* de ejecución de un TPA a una posible traza de ejecución de un TPA que empieza en el estado inicial y termina en un estado final que tiene asociado un tiempo de ejecución de inicio y de fin del estado. Estas muestras están formadas por las acciones y las transiciones entre los estados de esta.

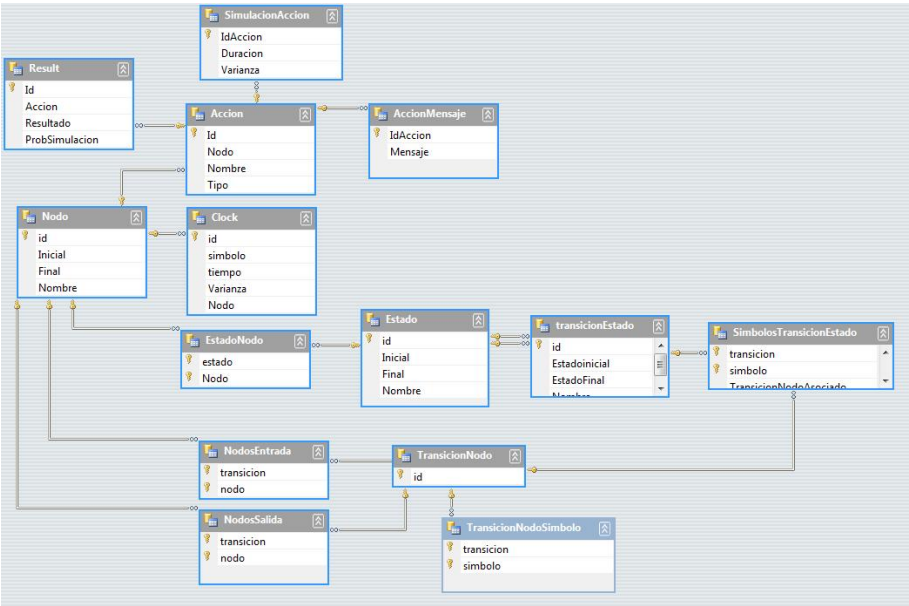

Figura 7.3: Esquema XML de la especificación de Workflow utilizado por el TPAEngine

7.3.4. TPA Engine WLogs

Una herramienta secundaria aunque importante del TPA Engine es el TPA Engine Log. El TPA Engine Log es el encargado de almacenar en un fichero de texto todas las muestras de ejecución de cada una de las instancias de WF. Esto no solo permite trazar la ejecución de los procesos en casos de error, si no que también puede ser utilizada para ayudar al diseño de procesos mediante la utilización de técnicas de WM. Un ejemplo de como se almacena la información en el log es mostrada a continuación

...
03-08-2007 14:59:43: i:0000 Nodo: Q1 ->InicioAccion: Q1.Q1
03-08-2007 14:59:43: i:0001 Nodo: Q0 ->InicioAccion: Q0.Q0
03/08/2007 14:59:48: i:0000 Fin Accion: Q1.Q1 Res: R11
03/08/2007 14:59:48: i:0000 Nodo: Q2 ->InicioAccion: Q2.Q2
03/08/2007 14:59:48: i:0002 Nodo: Q0 ->InicioAccion: Q0.Q0
03/08/2007 14:59:53: i:0001 Fin Accion: Q0.Q0 Res: KO
03/08/2007 14:59:53: i:0001 Instancia Finalizada: Estado Final X
03/08/2007 14:59:58: i:0000 Fin Accion: Q2.Q2 Res: R21

ËÏl TPA almacena la fecha y hora de la acción, la instancia asociada (i), el nodo asociado (Nodo), si el nodo se inicia o a finalizado y en el caso de haber finalizado, el resultado de la acción (Res). Como se puede ver, el TPA Engine Log, está diseñado para almacenar no solo la información eventual de las actividades, sino que también se almacenan el final y los resultados de estas. Esto permite al TPA Engine, que los procesos ejecutados en el pueden ser tratados posteriormente con técnicas de ABWM.

7.4. Aportaciones del TPA

El TPA Engine cuenta con grandes ventajas con respecto a muchos motores del mercado, como la eficiencia y la facilidad de ejecución del modelo. Se van analizar las ventajas del TPA Engine en las principales aplicaciones de este: La ejecución de WF, la simulación de procesos y la creación de corpus de ABWM.

- **Ejecución de WF con TPAs** Una de las aplicaciones para las que puede ser utilizada la interpretación de TPAs es la ejecución de procesos de negocio. El TPAEngine es un motor ligero capaz de manejar una gran cantidad de instancias de WF utilizando pocos recursos. Esto es debido a la excepcional sencillez del TPA. De hecho, en el proyecto Heart Cycle [27], Persona [30] y el proyecto PIPS [31] se ha hecho un estudio de los diferentes modelos de interpretación y se ha concluido que el la utilización de TPAs para ejecutar WF que requieran una alta expresividad es la solución más eficiente.

- **Simulación de Procesos de negocio con TPAs** Cuando se diseñan WF ya sea de forma tradicional, o mediante técnicas de WM. El proceso de prueba de los sistemas puede resultar complicado. Usualmente los procesos que requieren un gran paralelismo impiden a los diseñadores encontrar problemas en el diseño de WF como puedan ser ciclos infinitos, ramas sin salida o errores de sincronización. Estos errores normalmente son encontrados cuando el WF ya está implantado, y es necesario parar los procesos en ejecución y revisar la especificación para solucionar el problema.

  En este caso se puede utilizar el TPAEngine para simular la ejecución de WF. Esto va a permitir probar los WF antes de implantarlos en la vida real. Utilizando el TPAEngine se puede simular la ejecución de dos formas, una probando todos los caminos posibles, y la otra, ejecutando el WF y seleccionando aleatoriamente las transiciones que van a ser ejecutadas en cada derivación.

- **Generación de corpus para evaluar la inferencia de TPAs**

  Una vez ha sido implantado un WF es posible recoger los WLogs generados para crear un corpus con las acciones efectivamente realizadas. Esto apoyará la utilización de sistemas de WM en ciclos de diseño posteriores, ayudando a la mejora de los procesos en ejecución.

  Por otro lado, gracias a la capacidad de simulación del TPAEngine, es posible crear corpus de prueba que ayuden a evaluar los algoritmos de WM. El TPAEngine puede simular todas los caminos posibles de un WF y almacenar estos en un WLog. Este corpus puede ser usado para validar algoritmos de ABWM, evaluando las similitudes entre el WF Real y el WF inferido y probando las capacidades de inferencia de los algoritmos de WM para patrones específicos de WF.

7.5. Conclusiones y adecuación a la interpretación de Vías Clínicas

En el principio del capítulo se han estudiado las características que había de cumplir un motor de WF para la ejecución de Vías Clínicas. Estas características son; la *no limi-*

tación de la expresividad, la *modificación dinámica de instancias*, la creación de *WLogs compatibles con ABWM*, y la posibilidad de *Simulación de Procesos*

En cuanto a la expresividad, el TPAEngine es capaz de ejecutar los TPAs sin limitar su expresividad. La interpretación se ha abordado aprovechando el marco teórico del TPA para utilizar los modelos de estados finitos para ejecutar los procesos.

Para solucionar la modificación dinámica de las instancias, el TPAEngine almacena las junto con las instancias la información de flujo. Esta información inicialmente es copiada de la plantilla original del WF. Sin embargo, al estar la instancia totalmente desligada de la plantilla original dirigiendo su flujo a través de una copia, es posible modificarla en cualquier momento para realizar los cambios que fueran necesarios sin afectar al modelo original.

El WLogs del TPAEngine almacena no solo la información del inicio y del fin de las acciones realizadas en un WF. En este caso, el TPAEngine también enriquece la información del WLog con el resultado de las acciones realizadas. Esta es la información básica necesaria para poder realizar ABWM.

Al TPAEngine se le ha dotado de capacidad de simulación tanto programable, decidiendo que las probabilidades de ejecución, como canónica de modo que se pueden probar todas las derivaciones posibles del WF diseñado. Esto permite el probar de los WF antes de su implantación y la creación de corpus de aprendizaje para probar algoritmos de ABWM.

Capítulo 8

Algoritmo de Inferencia de Trazas basadas en Actividades Paralelas

La utilización de técnicas de ABWM para el apoyo al diseño de procesos de cuidado es una buena herramienta para ayudar en las iteraciones del diseño de Vías Clínicas. Este tipo de técnicas necesitan de algoritmos que utilicen muestras de ejecución pasadas en la inferencia de nuevos modelos de WF que expliquen mejor la realidad de los procesos. Este proceso se realiza de forma que es continuamente monitorizado por un experto en Vías Clínicas que valida los resultados del algoritmo para adecuarlos a la forma estándar de ejecutar el proceso.

Actualmente no existen algoritmos de WM capaces de aprovechar la información de los modelos de ABWM. En este capítulo se va a abordar la creación de un algoritmo que siga el modelo de ABWM, por lo que utiliza WLogs basados en actividades para inferir WF.

8.1. Aprendizaje de Flujos de Trabajo Orientado a Actividades basado en TPAs para el diseño de Vías Clínicas

Para conseguir estandarización de procesos en el campo de las Vías Clínicas usualmente los médicos utilizan la información hallada en los casos clínicos y en la experiencia personal para diseñar manualmente estas guías de actuación. El análisis de la información clínica es un complejo proceso que requiere de muchos recursos en forma de profesionales de la medicina expertos en determinados campos que, utilizando el consenso, describen los cuidados estandarizados para cada tipo de enfermedad. Aún así, estos procesos de cuidado estandarizados pueden no ser útiles en todos los casos. La variabilidad de las culturas, alimentación, legislaciones, patologías e incluso aceptación del paciente hace que las Vías Clínicas hayan de ser estudiadas en cada implantación para asegurarse que funciona en el entorno destino.

La utilización de tecnologías de WF mediante TPAs y técnicas de ABWM, pueden ayudar a la medicina a facilitar la introducción de las Vías Clínicas en el proceso de cuidado diario de los pacientes. Una Vía Clínica inicial puede ser implementada en forma de TPA para guiar el proceso de cuidado de una determinada enfermedad. Cuando esta Vía Clínica

es ejecutada para cada paciente, el médico puede cambiar el proceso en cualquier momento. Esto hace que la ejecución de los procesos sea distinta para cada paciente y la traza de ejecución quedará guardada en el WLog del TPAEngine. La utilización de un algoritmo de ABWM basado en TPA podría utilizar estas trazas para inferir un WF que describa mejor la realidad de una forma automática y acorde con las características del entorno donde esté implantado el sistema que pueda volver a ser utilizado por el TPAEngine. Sin embargo, dada la necesidad de ser estrictos en evitar errores el diseño de Vías Clínicas, el profesional de la medicina ha de mediar en el proceso de aprendizaje pudiendo cambiar en cualquier momento el TPA si no se adecúa suficientemente al problema. Por ello, al realizar un proceso de ABWM, el TPA es presentado al médico y es susceptible de ser cambiado para eliminar errores en el aprendizaje y mejorar el TPA de la iteración anterior.

Para poder realizar este proceso, es necesario obtener un algoritmo de ABWM que tenga las siguientes características:

- **Utilización de TPAs**.- La utilización de TPA permite hacer uso del aprendizaje de WF mediante ABWM. Además, la utilización de TPAs para el aprendizaje tiene la ventaja de que existen gran cantidad de marcos teóricos formales para el aprendizaje de modelos de estados finitos regulares por lo que existen técnicas de gran potencia en la literatura que podrían ser utilizadas para aprender TPAs. Además, el TPA tiene una buena expresividad, por lo que puede ser usado para ser directamente el modelo de representación de Vías Clínicas sin tener que traducirlo a un segundo lenguaje.

- **Aprendizaje de Procesos Paralelos**.- Para las Vías Clínicas es crucial la representación de procesos paralelos. Es muy común que en los procesos de cuidados se realicen acciones paralelas incluso realizadas por profesionales diferentes.

- **Aprendizaje basado en actividades**.- Para que los modelos de WF aprendidos sean capaces de representar la realidad con cierta utilidad, es necesario que los algoritmos de aprendizaje utilicen la información relativa a los resultados de las actividades y la incorporen al WF en forma de condición de transición entre los estados.

Aunque existen algoritmos de WM que aprenden procesos paralelos, desgraciadamente, en la literatura no existen algoritmos de WM que aprendan TPAs, y mucho menos que lo hagan dentro de la filosofía del ABWM. Por esta razón, en este capítulo se va a presentar un algoritmo de ABWM cuyo modelo de representación sea el TPA capaz de inferir procesos paralelos.

8.2. Algoritmo de Inferencia de Trazas basadas en Actividades Paralelas

La utilización del modelo formal del TPA como objeto de representación para modelos de ABWM permite el uso de técnicas clásicas de reconocimiento de formas que han sido utilizadas con éxito en otros campos. Entre ellas, la *Inferencia Gramatical* [55]. La Inferencia Gramatical es una disciplina del reconocimiento de formas que se basa en la utilización de técnicas de Reconocimiento Sintáctico de Formas para el aprendizaje de

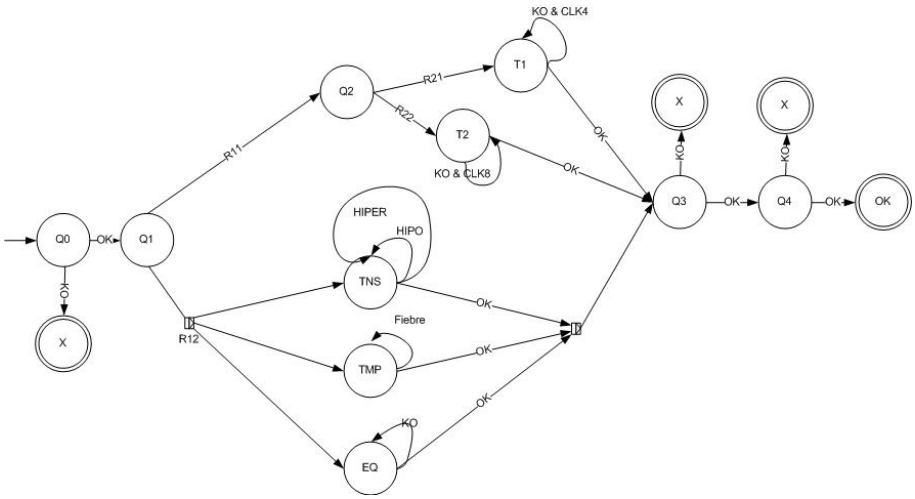

Figura 8.1: Ejemplo de Workflow a inferir por PALIA

gramáticas formales a partir de muestras de entrada. Algoritmos como OSTIA (del inglés; *Onward Subsequential Transducer Inference Algorithm*) [79] que aprende transductores para su uso en sistemas de traducción automática; ESMA [35] utilizado para inferir clasificadores con autómatas finitos; o ALERGIA [78] pensado para inferir lenguajes regulares, utilizan la Inferencia Gramatical para aprender modelos de estados finitos para resolver diversos problemas.

En esta línea se presenta un algoritmo de ABWM basado en técnicas de inferencia gramatical capaz de inferir TPAs a partir de muestras de entrada llamado PALIA. PALIA (del inglés *Parallel Activity-based Log Inference Algorithm*) o Algoritmo de Inferencia de Trazas basadas en Actividades Paralelas utiliza *técnicas clásicas de inferencia gramatical adecuándolas al aprendizaje de TPAs*. PALIA es capaz de *aprender modelos basados en actividades* ya que utiliza para el aprendizaje corpus de entrada basados en actividades. En concreto PALIA utiliza como entrada muestras en el formato del WLog del TPAEngine. Además, PALIA infiere *procesos paralelos* y los representa en forma de TPA.

El algoritmo PALIA está dividido en distintas fases. Cada una de las fases toma el resultado de la anterior para inferir el WF asociado. Estas fases son las siguientes:

- **Árbol Aceptor de Prefijos Paralelo**.- En esta fase, el algoritmo toma como entrada un WLog donde se encuentra la información relativa a la ejecución de las actividades y devuelve un TPA que representa la muestra de entrada. En este proceso se analizan las muestras de entrada y se identifican las acciones que se ejecutan en paralelo, que son aquellas cuya intersección temporal es distinta de cero.

- **Merge Paralelo**.- En esta fase, el algoritmo toma como entrada el árbol aceptor de prefijos y produce como salida un TPA generalizado. El Merge paralelo produce generalizaciones mediante la fusión de nodos y transiciones sucesivas equivalentes identificando las secuencias tanto paralelas como no paralelas. En esta fase, cada nodo del árbol identifica sus nodos siguientes equivalentes fusionándolos. Además, en el caso de las secuencias paralelas, identifica que nodo siguiente pertenece a cada

secuencia atendiendo a la información temporal de las ejecuciones de las acciones excluyendo a dichas acciones de las secuencias que se intersecten temporalmente con ellas.

- **Onward Merge**.- En esta fase se toma como entrada la salida del Merge paralelo y devuelve un TPA generalizado. En esta fase se detectan las sincronizaciones de las secuencias tanto paralelas como no paralelas atendiendo a la repetición de las ramas equivalentes. Para ello se fusionan las ramas formadas por nodos y transiciones que sean equivalentes.

- **Eliminación de transiciones repetidas y nodos inútiles**.- En esta fase toma como entrada el resultado del algoritmo Onward Merge y devuelve un TPA sin transiciones repetidas ni estados inútiles. Esta fase se encarga de analizar el TPA de la entrada y eliminar toda aquella información redundante o inútil que empobrece la legibilidad del TPA.

A continuación se presentan con detalle cada una de las fases. Para ilustrar el funcionamiento del algoritmo se ha escogido un ejemplo sencillo que se muestra en la Figura 8.1. Este ejemplo simple cuenta con separaciones y sincronizaciones paralelas de los nodos TNS, TMP y EQ. Además, cuenta con elecciones exclusivas (XOR) y sincronizaciones simples entre T1 y T2.

8.2.1. Árbol Aceptor de Prefijos Paralelo

El árbol aceptor de prefijos paralelo toma como entrada un corpus de muestra de ABWM en el formato de WLog del TPAEngine. El resultado final esperado de este algoritmo es un TPA que acepta la muestra de entrada.

La entrada del algoritmo es el conjunto de muestras de WF M. Cada muestra representa una ejecución pasada de WF. La instancia contiene cronológicamente ordenadas las acciones realizadas. Un ejemplo se muestra se puede ver a continuación:

...
18/09/2007 13:19:28 =>i:0000 Nodo: Q0 ->InicioAccion: Q0.Q0
18/09/2007 13:19:30 =>i:0000 Fin Accion: Q0.Q0 Res: OK
18/09/2007 13:19:30 =>i:0000 Nodo: Q1 ->InicioAccion: Q1.Q1
18/09/2007 13:19:31 =>i:0000 Fin Accion: Q1.Q1 Res: R11
18/09/2007 13:19:31 =>i:0000 Nodo: Q2 ->InicioAccion: Q2.Q2
18/09/2007 13:19:32 =>i:0000 Fin Accion: Q2.Q2 Res: R21
18/09/2007 13:19:32 =>i:0000 Nodo: T1 ->InicioReloj: $CLKT1_4$
18/09/2007 13:19:32 =>i:0000 Nodo: T1 ->InicioAccion: T1.T1
18/09/2007 13:19:33 =>i:0000 Fin Accion: T1.T1 Res: OK
18/09/2007 13:19:33 =>i:0000 Nodo: T3 ->InicioReloj: $CLKT3_48$
18/09/2007 13:19:37 =>i:0000 ->FinReloj: $CLKT1_4$
18/09/2007 13:19:39 =>i:0000 ->FinReloj: $CLKT3_48$
18/09/2007 13:19:39 =>i:0000 Nodo: Q3 ->InicioAccion: Q3.Q3
18/09/2007 13:19:40 =>i:0000 Fin Accion: Q3.Q3 Res: OK
18/09/2007 13:19:40 =>i:0000 Nodo: Q4 ->InicioAccion: Q4.Q4
18/09/2007 13:19:41 =>i:0000 Fin Accion: Q4.Q4 Res: OK
18/09/2007 13:19:41 =>i:0000 Instancia Finalizada: Estado Final F
...

El algoritmo presentado en esta sección es una modificación del árbol aceptor de prefijos clásico clásico permitiendo la definición de ramas paralelas que representan la ejecución simultanea de varias acciones. Formalmente:

Definición 8.1 *Dada una muestra s del corpus S de define formalmente el Árbol Aceptor de Prefijos Paralelo(PPTA) como:*
PPTA(s) = {I, F, γ,δ},

$I = \{\lambda\}$, *Si* $s = \emptyset$ *entonces* $I = \emptyset$

$F = s$,

$\delta(q_u, Res_{q_u}q_a) = q_uq_a$, *siendo* q_u, q_a *y* $q_uq_a \in Q$

$\gamma(\{u_0..u_n\}, Res_{u_0..u_n}\{a_0..a_m\}) = \{u_0..u_n\}\{a_0..a_m\}$, *siendo* $\{u_0..u_n\} \in q_u$ *y* $\{a_0..a_m\} \in q_a$

Un PPTA es una estructura que acepta la muestra de entrada. Este modelo aplica dos funciones sobre la muestra de entrada,La función δ define las función de los estados en un TPA. Por otro lado la función γ representa la función de nodos de un TPA. Los estados representan acciones secuenciales por lo que en ningún caso la función δ tendrá transiciones de muchos a muchos. Sin embargo, la función de nodo puede tener transiciones de muchos a muchos especificando así el paralelismo.

Los nodos se van creando conforme a la ejecución de acciones, estos nodos son asignados a estados. Cuando llega un nuevo nodo que es paralelo al anterior, este se añade al mismo estado. Esto se repite hasta que llega un nodo que no es paralelo con el anterior. En ese momento se crea un nuevo estado al que se le añade un nuevo nodo. A partir de este instante se crean las transiciones de δ, que simplemente es una transición entre el estado anterior y el nuevo estado creado, y las transiciones de γ, que son transiciones entre todos los nodos del estado anterior, y todos los nodos que se encuentren en el nuevo estado. □

A continuación vamos a definir formalmente el concepto de *Acción* y *Acción Paralela* que son necesarios para la comprensión del algoritmo.

Definición 8.2 *Dado un TPA* $A = \{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$ *y el conjunto* $T_A \subseteq Muestras(A)^*$;

$Accion(A, T_A) = \{a_0, a_1, .., a_i, .., a_n \ ; 0 \leq i \leq n / \ a_i = (n, \{i_0, i_1, .., i_j, .., i_m\})$ *donde* $n \in N$
$\wedge i_j = (q_j, a_j, q_{j+1}, t, t + \alpha) \in Transiciones(T_{A_k}) \wedge \forall i_j; n \in q_j\}$

Las muestras de ejecucion contienen información sobre las acciones. Se denomina *Acción* de un TPA y asociado a un conjunto de Muestras a cada uno de los procesos que puede realizarse de forma atómica junto con sus ejecuciones efectivas. El concepto de acción es equivalente al concepto del Nodo en un TPA solo que la acción incorpora la información temporal de todas sus ejecuciones. □

Definición 8.3 *Dadas dos acciones* $a_0 = \{n_0, \{h_0, h_1, .., h_i, .., h_n\}\}$ *y* $a_1 = \{n_1, \{k_0, k_1, .., k_j, .., k_m\}\}$ *se dice que son paralelas si* $\exists h_i = (q_{h_i}, a, q_{h_{i+1}}, t_x, t_y) \in a_0 \wedge k_i = (q_{k_j}, a, q_{k_{j+1}}, t_v, t_w) \in a_1 \ / \ t_v < t_x < t_w \vee t_v < t_y < t_w$

Intuitivamente se dice que una acción es *Paralela* a otra cuando la intersección de sus ejecuciones temporales es distinto de vacio. □

El algoritmo 8.2.1 primero inicializa un TPA que será el resultado de este algoritmo. A continuación, para cada una de las muestras de entrada, identifica el estado inicial, que será el conjunto de acciones paralelas que tienen lugar al iniciarse la instancia de WF de la muestra. A partir de este momento se va comparando la ejecución de la muestra con el árbol aceptor de prefijos paralelo inferido hasta el momento, y se van añadiendo los nodos con las acciones que no son admitidas por este. Para ello se van analizando una a una las acciones de la muestra en el TPA creando las transiciones que fueran necesarias. El último estado de la muestra se declara como final.

Algoritmo 8.1

```
ÁRBOL ACEPTOR DE PREFIJOS PARALELO(M)
 1  TPA ← INICIALIZARTPA()
 2  for m ∈ M
 3        do
 4           Ini ← GetEstadoInicial(TPA, m)
 5           t.AddEstadoInicialIni
 6           ▷ t es la Transición actual
 7           for acc ∈ m ▷ para cada Nodo de la Muestra m
 8                 do
 9                    estado ← CreaSiguienteTransicion(TPA, t, acc)
10           CrearEstadoFinal(TPA,t.EstadoFinal)
11  return TPA
```

En el algoritmo 8.2 se puede ver con más detalle el paso de crear los nuevos nodos y transiciones acordes con las nuevas muestras en el algoritmo. Este algoritmo crea una

transición nueva desde la transición actual de la muestra que está siendo analizada. La entrada del algoritmo es el TPA actual, la transición actual y la acción que representa el nuevo nodo que hay que añadir al TPA. El algoritmo primero calcula si las acción es paralela históricamente con el rango o el dominio de la transición actual, es decir, que en todos los casos estas acciones se han ejecutado en paralelo. En este caso, la acción es añadida al rango o al dominio respectivamente. Si la acción no es paralela históricamente ni al dominio, ni al rango de la transición, se crea una nueva transición cuyo dominio sea el rango de la transición actual, y el rango sea la acción.

Algoritmo 8.2

CreaSiguienteTransicion(TPA,t,Nodo)

```
1  if accionParalelaEstado(Nodo, t.Dominio)
2      then t.AddAccionaDominio(Nodo)
3           return t
4  if accionParalelaEstado(Nodo, t.Rango)
5      then t.AddAccionaRango(Nodo)
6           return t
7  TPA.AddTransicion(t)
8  t.Dominio ← t.Rango
9  t.Rango ← ∅
10 t.AddNodoaRango(Nodo)
11 return t
```

El aspecto gráfico que toma el WF final es parecido a un árbol como puede verse en la Figura 8.2. El árbol de la figura acepta únicamente las muestras de entrada. Cuando no existe ninguna acción paralela, el algoritmo funciona exactamente igual que el modelo clásico: explorando linealmente el árbol, y creando nuevos nodos si es preciso. En el caso en que se encuentre un conjunto de acciones paralelas las trata como un único nodo, y genera un transiciones paralelas a este conjunto de nodos.

8.2.2. Algoritmo Merge Paralelo

El aprendizaje realizado con el árbol aceptor de prefijos inicial solo ha servido para aprender la muestra. Los siguientes pasos consisten en generalizar la información para obtener modelos más adecuados.

En esta línea, el Algoritmo Merge Paralelo toma como entrada el árbol aceptor que es salida del algoritmo anterior y realiza fusiones de transiciones y nodos equivalentes. El algoritmo 8.3 especifica el Merge Paralelo de un modo general. Este algoritmo recorre todas las transiciones del árbol para juntar los nodos que sean equivalentes. El algoritmo trata de distinta manera las transiciones simples (las que van de nodo a nodo), las transiciones de rango simple (las que van de un nodo a varios nodos) y las transiciones de rango complejo (las que van de varios nodos a varios nodos).

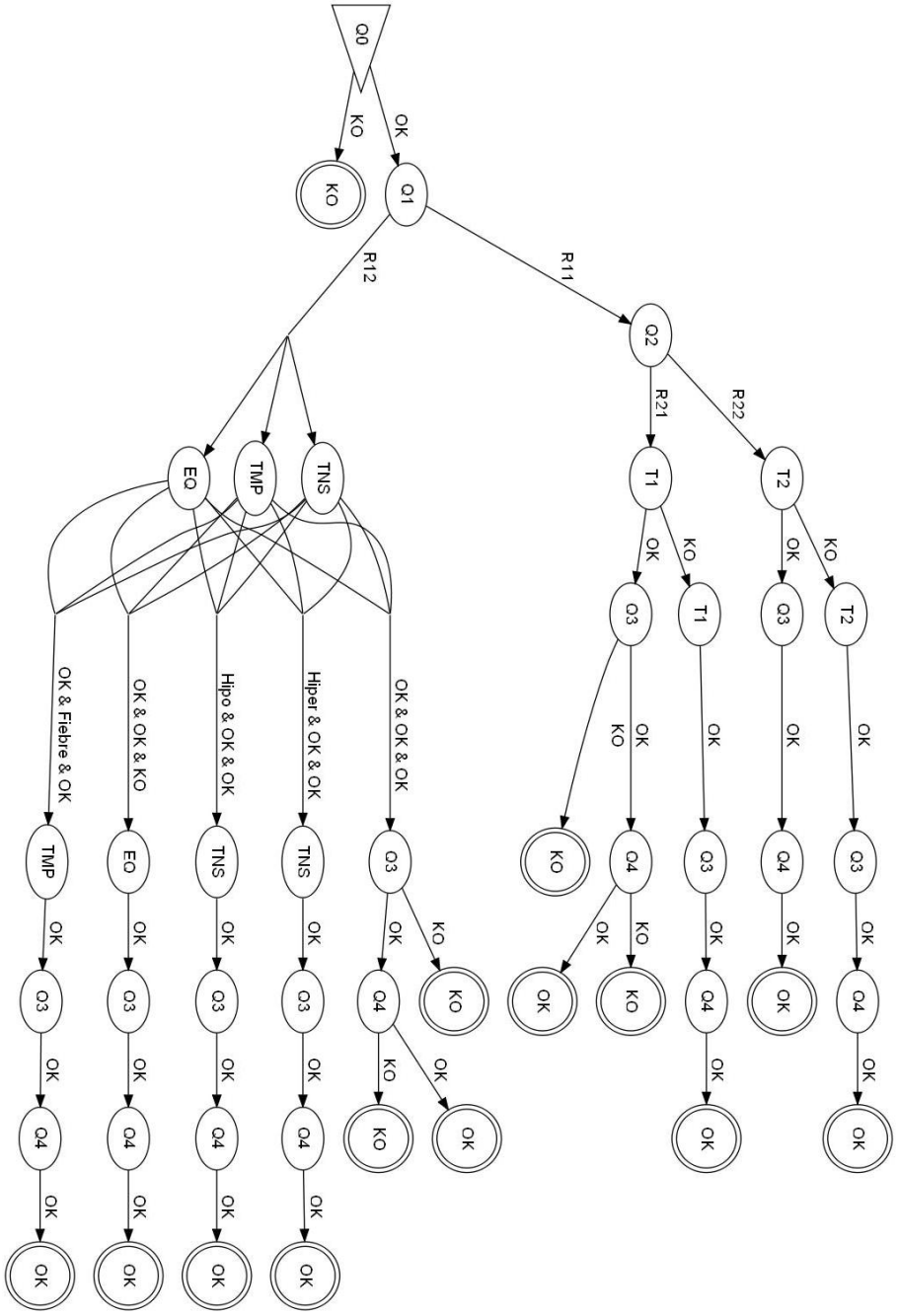

Figura 8.2: Resultado del algoritmo Árbol Aceptor de Prefijos

Algoritmo 8.3

```
MergeTransicion(TPA)
 1  for t ∈ TPA.Transiciones
 2        do
 3           if t.Dominio.Count = 1&t.Rango.Count = 1
 4              then MergeTransicionSimple(t)
 5                   return
 6           if t.Rango.Count = 1
 7              then MergeTransiciondeRangoSimple(t)
 8                   return
 9           if t.Rango.Count > 1
10              then MergeTransiciondeRangoParalelo(t)
11                   return
```

El algoritmo de forma general busca transiciones y acciones que resultan equivalentes para fusionarlas o paralelizarlas en su caso. A continuación se definen los conceptos de *Acciones Equivalentes* , *Fusión de Acciones*, *Acciones Previas Históricas* y *Estados Previos Históricos*.

Definición 8.4 *Dadas dos Acciones a_0 y a_1 de un TPA A se dice que $a_0 \equiv a_1$ si $n_0 \in a_0 = n_1 \in a_1$*

Dadas dos Transiciones t_0 y t_1 de un TPA A se dice que $t_0 \equiv t_1$ si $Rango(t_0) = Rango(t_1) \wedge Dominio(t_0) = Dominio(t_1)$

Se dice que dos Acciones son *Acciones Equivalentes($\equiv$)* cuando tienen asignados los mismos nodos. De forma análoga, Se dice que dos transiciones son *Transiciones Equivalentes($\equiv$)* cuando sus rangos y sus dominios lo son. □

Definición 8.5 *Dadas dos acciones equivalentes acc_0 y acc_1*

$Fusion(acc_0, acc_1) = \{n, \{i_{0_0}, .., i_{0_n}, i_{1_0}, .., i_{1_m}\}\}$ *donde* $n \in acc_0, acc_1 \wedge \{i_{0_0}, .., i_{0_n}\} \in acc_0 \wedge \{i_{1_0}, .., i_{1_m}\} \in acc_1$

Dadas dos acciones equivalentes acc_0 y acc_1, se denomina **Fusión de Transiciones** de dos acciones al proceso consistente en eliminar la acción acc_1 y sustituir del dominio de todas las transiciones acc_1 por acc_0. □

Además el algoritmo puede encontrar patrones de paralelizacion entre acciones buscando estados y acciones previas históricas.

Definición 8.6 *Dadas dos acciones $acc_i = \{n_0, \{i_0, .., i_k, .., i_n\}\}$ y $acc_j = \{n_1, \{j_0, .., j_l, .., j_m\}\}$ donde $0 \leq k \leq n$ y $0 \leq l \leq m$, se dice que $acc_0 \prec acc_1$ si $\nexists i_k = (q_{h_i}, a, q_{h_{i+1}}, t_x, t_y) \wedge j_l = (q_{k_j}, a, q_{k_{j+1}}, t_v, t_w) \mid t_v < t_x < t_w \vee t_v < t_y$*

Dados $c_a = \{a_0, .., a_i, .., a_n\}$ y $c_b = \{b_0, .., b_j, .., b_m\}$ se dice que $c_a \prec c_b$ si $\forall a \in c_a \wedge b \in c_b | a \prec b$

Dadas dos acciones acc_0 y acc_1, se denomina que son *Acciones Previas Históricas* ($\prec$) cuando no existe ninguna ejecución de la acción acc_0 que haya sido ejecutada de forma paralela o posterior con ninguna ejecución de la acción acc_1.

De forma análoga, dados dos conjuntos acciones c_0 y c_1, se denomina que son *Estados Previos Históricos*($\prec$) cuando todas las acciones pertenecientes al estado c_0 son acciones previas históricas de todas las acciones del estado c_1 □

Algoritmo 8.4

```
MergeTransicionSimple(t)
1  if t.NodoInicial ≡ t.NodoFinal
2     then FusionaEjecuciones(t.NodoInicial,t.NodoFinal);
3          EliminarTransicion(t);
4          FusionaTransiciones(t.NodoInicial,t.NodoFinal);
5          CrearTransicion(t.NodoInicial,t.NodoInicial);
```

El Algoritmo 8.4 muestra la fusión de transiciones simples. Para cada transición simple se comprueba si el nodo origen es equivalente al nodo destino. En ese caso se fusionan los nodos eliminando la transición y se creando otra transición del nodo origen a si mismo. Además todas las transiciones cuyo dominio sea la acción rango de la transición eliminada son modificadas para que el dominio de esta sea igual al dominio de la transición eliminada. Por otro lado las ejecuciones asociadas al nodo rango son asociadas al nodo dominio.

Algoritmo 8.5

```
MergeTransicionDeRangoSimple(t)
 1  if {∃acc0, acc1 ∈ t.Dominio|acc0 ≡ acc1}
 2     then FusionaEjecuciones(acc0,acc1);
 3          EliminarAccionDeTransicion(t, acc1);
 4          FusionaTransiciones(acc0,acc1);
 5          return
 6  if {∃acc ∈ t.Dominio|acc ≡ t.AccionFinal}
 7     then FusionaEjecuciones(acc,t.AccionFinal);
 8          FusionaTransiciones(acc,t.AccionFinal);
 9          EliminarTransicion(t);
10          CrearTransicion(acc,acc);
11          return
12  if {∃acc ⊆ t.Dominio|acc ≺ t.AccionFinal}
13     then CrearTransicion(acc,t.AccionFinal);
14          CrearTransicion(t.Dominio-acc+t.AccionFinal,
15          Rango(t.AccionFinal))
16          EliminarTransicion(t);
17          return
```

El Algoritmo 8.5 se encarga de la fusión de las transiciones que tienen dos o más nodos

en dominio y un nodo en rango. En este algoritmo, si el nodo rango es equivalente alguno de los nodos del dominio de la transición este nodo se fusionará del mismo modo que una transición simple. Por otro lado, si en el dominio hay algún par de acciones equivalentes, estas se fusionan. Por último si hay algún subconjunto del dominio de la transición de acciones históricamente previas al nodo rango de la transición, esta transición se divide en dos: una que va desde este subconjunto al nodo rango, y otra que va desde todo el dominio al rango del nodo rango de la transición

Por último, el Algoritmo 8.6 se encarga de las transiciones no contempladas en los anteriores algoritmos. Del mismo modo que en el algoritmo 8.5, en el caso en el que el dominio de la transición contenga algún par de acciones equivalentes, estas se fusionan. Si en el dominio existe algún nodo equivalente a algún nodo del rango de la transición, el algoritmo fusionaría las ejecuciones crearía una transición del resto del dominio al rango de la acción y fusionaría las ejecuciones. Por último, si existe un subconjunto del dominio que sea históricamente previo a un subconjunto del rango, se crearía una transición entre ellos y otra que fuera desde el resto del dominio más el subconjunto históricamente previo al rango de este.

Algoritmo 8.6

```
MergeTransicionDeRangoParalelo(t)
 1  if {∃acc0, acc1 ∈ t.Dominio|acc0 ≡ acc1}
 2      then FusionaEjecuciones(acc0,acc1);
 3           EliminarAccionDeTransicion(t, acc1);
 4           FusionaTransiciones(acc0,acc1);
 5           return
 6  if {∃prev ∈ t.Dominio ∧ ∃sig ∈ t.Rango|prev ≡ sig}
 7      then
 8           CrearTransicion(t.Dominio-prev,Rango(sig))
 9           for acc0 ∈ prev
10                do
11                   if acc1 ∈ sig|acc0 ≡ acc1
12                   FusionaEjecuciones(acc0, acc1);
13                   FusionaTransiciones(acc0,acc1);
14                   CrearTransicion(acc0,acc0);
15                   return
16           EliminarTransicion(t);
17  if {∃prev ⊆ t.Dominio ∧ ∃sig ⊆ t.Rango|prev ≺ sig}
18      then CrearTransicion(prev,sig);
19           CrearTransicion(t.Dominio-acc+sig, Rango(sig))
20           EliminarTransicion(t);
21           return
```

El resultado del algoritmo se puede ver gráficamente con el ejemplo de la Figura 8.3. En este ejemplo se pueden ver como se han identificado los ciclos de acciones que resultaban equivalentes tanto en el caso de las transiciones simples (T2 y T1) como en el caso de las transiciones paralelas (TNS, EQ y TMP). En cuanto a las transiciones simples, existen

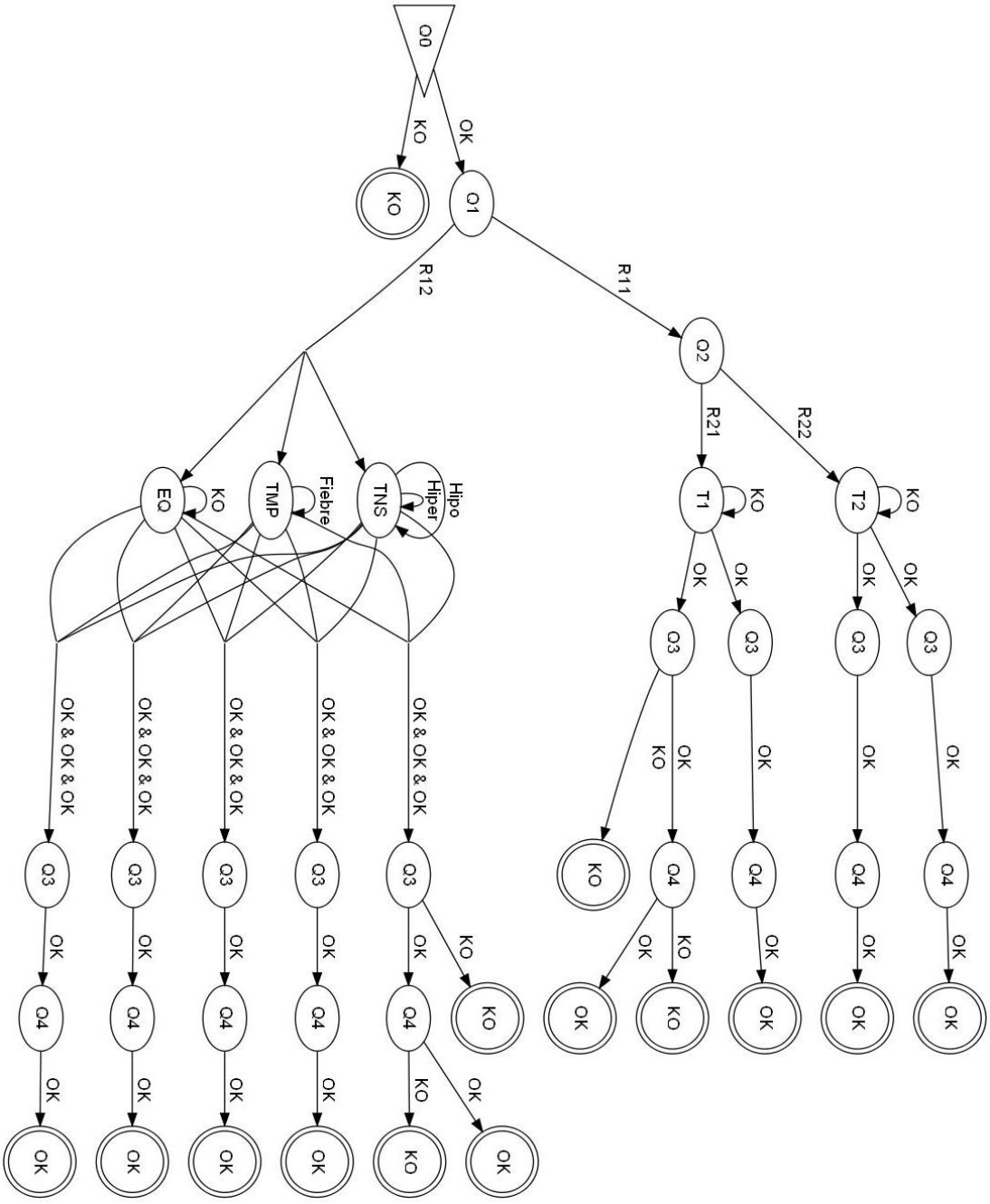

Figura 8.3: Resultado del algoritmo Merge Paralelo

acciones equivalentes siguientes a T1 y T2 que se fusionan con estas dando lugar a los ciclos. En el caso de las transiciones paralelas existen nodos previamente historicos y equivalentes a los nodos EQ, TNS y TMP que se fusionan a estas.

8.2.3. Algoritmo Onward Merge

El algoritmo Onward Merge utiliza como entrada el TPA salida del algoritmo del Merge Paralelo. El Onward Merge se encarga de fusionar ramas equivalentes de TPA. El Algoritmo 8.7 expresa en pseudocódigo el funcionamiento del algoritmo. El Onward Merge recorre todos los pares de acciones del TPA de modo que si son equivalentes y fusionables hacia adelante se fusionarían todos los nodos y transiciones de sus ramas. La definición de *Fusión hacia adelante* es la siguiente:

Definición 8.7 *Dadas dos Instancias* $I_i = i_0 i_1..i_k..i_n$ *y* $I_j = j_0 j_1..j_l..j_m$ *se dice que dos acciones* i_k *y* j_l *son Fusionables hacia adelante si* $\forall i_x \in i_k i_{k+1}..i_n \wedge j_x \in j_l j_{l+1}..j_m$ *donde* $0<x<z \wedge |i_k i_{k+1}..i_n| = |j_l j_{l+1}..j_m| = z$; $i_x = \{q_x, a_x, q_{x+1}, t_i, t_i + \alpha\} \wedge j_x = \{q_x, a_x, q_{x+1}, t_j, t_j + \alpha\}$

Dadas dos acciones acc_i y acc_j, se dice que son *Fusionables hacia adelante* cuando para cada una de las derivaciones de acc_i que lleve a un estado final existe una derivación equivalente a partir de acc_j y viceversa. □

Algoritmo 8.7

```
OnwardMerge(TPA)
1  for acc0, acc1 ∈ TPA
2        do
3           if (NodoFusionablehaciaAdelante(acc0,acc1))
4              then FusionaEjecuciones(acc0, acc1);
5                   FusionaTransiciones(acc0,acc1);
6                   EliminaNodo(acc1);
```

El Algoritmo 8.8 compara dos nodos para comprobar si son fusionables hacia adelante. Para ellos compara todas las transiciones cuyo dominio sean los nodos y las compara. Por recursividad va comprobando las ramas de los nodos hasta que se lleguen a nodos finales, o hasta que uno de los nodos no sean equivalentes.

Algoritmo 8.8

```
NodoFusionablehaciaAdelante(acc0,acc1)
 1  T0 ← getTransicionesporDominio(acc0)
 2  T1 ← getTransicionesporDominio(acc1)
 3  if ∀t0 ∈ T0∃t1 ∈ T1|t0 ≡ t1
 4     then
 5          for t0, t1 ∈ T0, T1|t0 ≡ t1
 6              do
 7                  if (NodoFusionablehaciaAdelante(t0.Rango,t1.Rango))
 8                     then FusionaEjecuciones(t0.Dominio, t1.Dominio);
 9                          FusionaTransiciones(t0.Dominio,t1.Dominio);
10                          EliminaNodo(t1.Dominio);
11                          return Si;
12                     else
13                          return No;
```

En la Figura 8.4 se pueden ver como se han fusionado las ramas. Se puede ver como las ramas referentes a las acciones T1 y T2 se han sincronizado en el estado Q3, al igual que las acciones paralelas TNS, TMP y EQ. Las ramas finales de todas estas acciones ha resultado ser equivalentes, y por ello se ha producido su fusión.

8.2.4. Algoritmo de Eliminación de transiciones repetidas y nodos inútiles

Para finalizar, se realiza una fase con un algoritmo que elimine todos las transiciones repetidas y los nodos inútiles. Las transiciones repetidas son aquellas que van de los mismos nodos a los mismos nodos con los mismos símbolos. Por otro lado, los nodos inútiles que no son finales y no tienen transición de salida, los que desde ellos no se puede llegar a ningún estado final o los que no son alcanzables desde ningún estado inicial. Además de que estos nodos no aportan nada al resultado final, son contraproducentes, ya que afectan sobre la legibilidad del TPA ya que aumenta la talla del resultado.

En la Figura 8.5 se ve el TPA resultante de este algoritmo. Se puede ver como al eliminar todas las transiciones repetidas obtenemos en WF original.

8.3. Implementación de PALIA

Para la pruebas se ha implementado un software utilizando el algoritmo PALIA para inferir WF utilizando como entrada corpus de muestras basados en TPA. PALIA se ha implementado en CSharp utilizando la plataforma .NET. La implementación realizada de PALIA está diseñada para aceptar el formato de WLog producido por el TPA Engine presentado anteriormente, y produce TPAs en el formato de entrada del TPAEngine. Además, el algoritmo produce una salida en dot y dibuja el WF inferido en pantalla.

En la Figura 8.6 se muestra una captura de pantalla de la aplicación realizada para la inferencia de WF

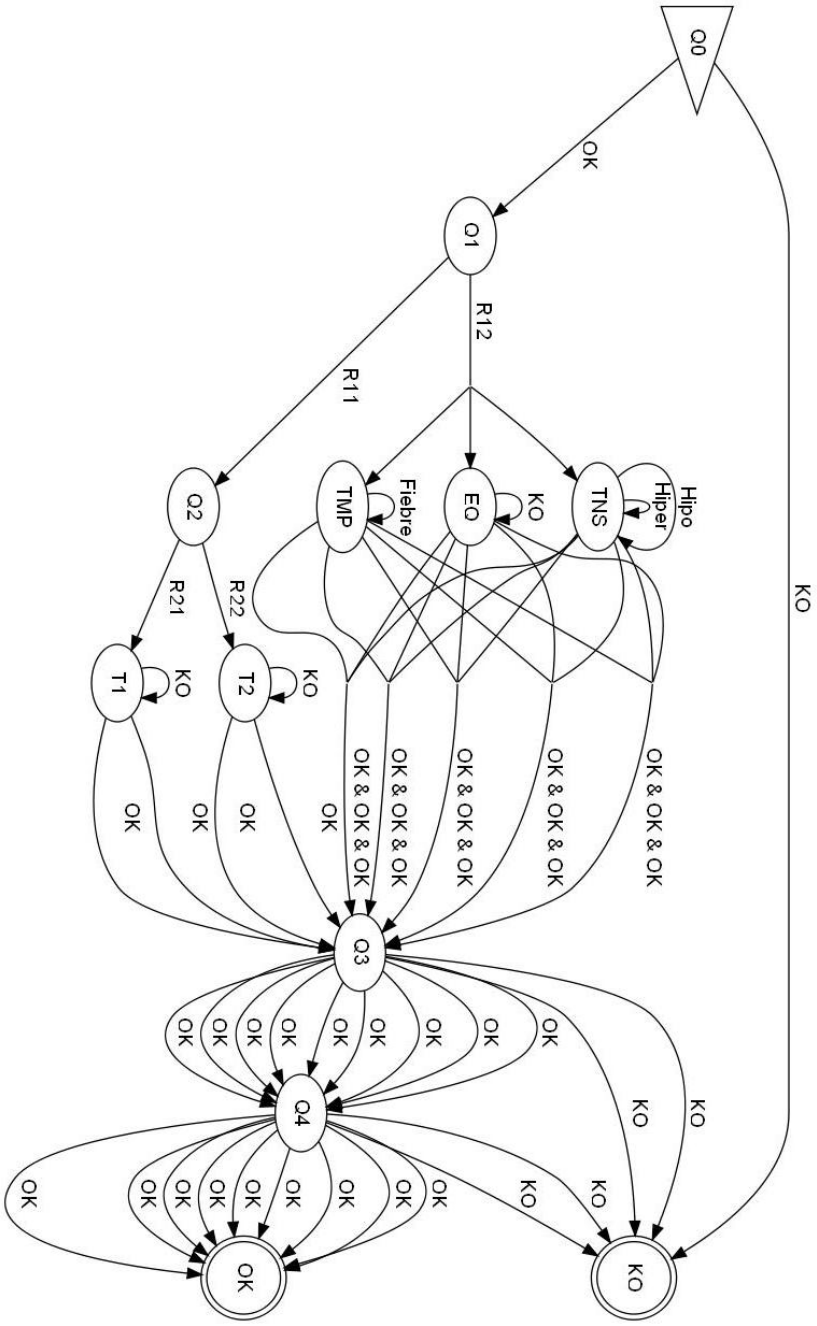

Figura 8.4: Resultado del algoritmo Onward Merge

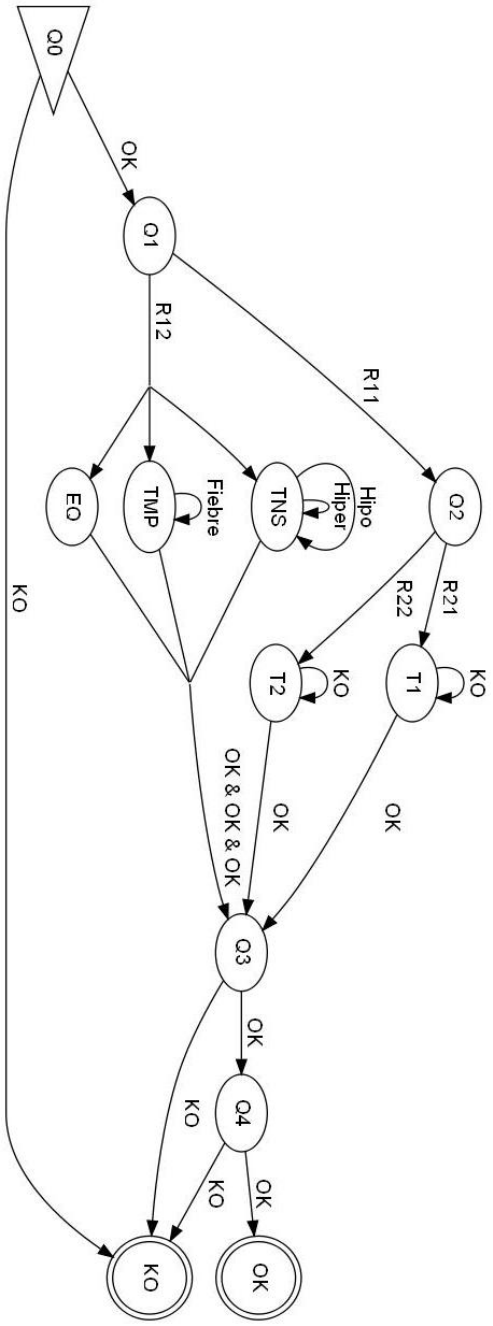

Figura 8.5: Resultado del algoritmo de eliminación de transiciones repetidas

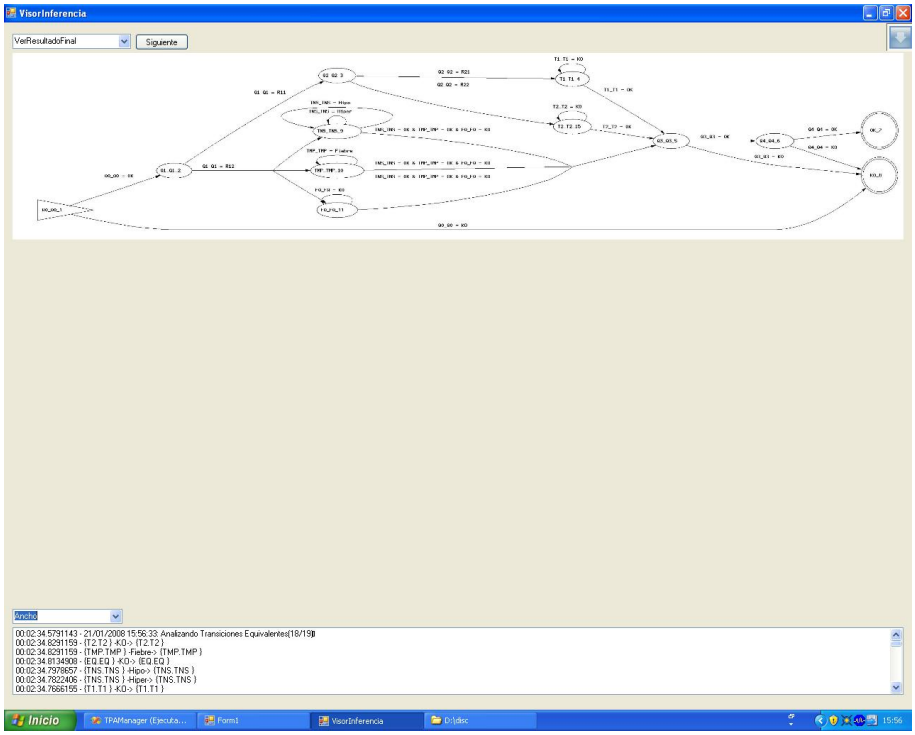

Figura 8.6: Software de control del algoritmo PALIA

Utilizando esta aplicación se pueden ejecutar PALIA sobre cualquier archivo de texto que esté en el formato adecuado. Además la aplicación puede ser configurada para que vaya mostrando los resultados de las diferentes fases.

El algoritmo PALIA ha sido presentado en [16].

8.4. Conclusiones y Adecuación de PALIA al aprendizaje de Vías Clínicas

En el principio del capítulo se especificaron las características que debía tener un algoritmo de WM para poder ser aplicado para apoyar el diseño de las Vías Clínicas. Las características que se destacaron fueron: *Aprendizaje de TPAs*, *Aprendizaje de Procesos Paralelos* y *Aprendizaje basado en Actividades*.

PALIA es un algoritmo totalmente basado en aprender TPA. No obstante, cada una de sus fases obtiene un TPA generalizado.

Por otro lado, PALIA es capaz de aprender procesos paralelos, ya que sus fase de creación del Arbol Aceptor y la fase del Merge tiene en cuenta las acciones que se ejecutan de forma paralela generalizando en consecuencia.

PALIA aprende un TPA incorporando los resultados de las acciones en las transiciones. Esto le hace capaz de realizar aprendizaje basado en Actividades.

En conclusión, PALIA ha satisfecho las condiciones iniciales, y resulta un algoritmo viable para ser usado como apoyo al diseño de Vías Clínicas. Una vez creado el algoritmo, el siguiente paso es validarlo para comprobar su efectividad en el siguiente capítulo se evaluará su efectividad con una serie de experimentos.

Capítulo 9

Resultados Experimentales

En este capítulo el algoritmo de WM presentado en el capítulo anterior será evaluado para determinar su eficacia. Esta eficacia será comparada con la de otros algoritmos de WF existentes en la literatura. Esta evaluación será efectuada en términos de eficacia y legibilidad. De este modo no solo se evaluará la capacidad del algoritmo para descubrir WF sino también su capacidad de representarlos.

9.1. Introducción

Después de haber presentado el algoritmo PALIA, el siguiente paso es la evaluación de su comportamiento. La medida de la bondad de los algoritmos de WM es un problema difícil de resolver. En la literatura no existen métricas estandarizadas que permitan la evaluación de este tipo de modelos. Es por esto que la evaluación de estos modelos varían de unos algoritmos a otros dependiendo de las cualidades de los propios sistemas de inferencia.

A pesar de esto, podemos encontrar un punto de convergencia entre los objetivos de los distintos algoritmos: todos buscan inferir modelos que identifiquen el mayor número de patrones de WF de la manera más legible posible.

Por un lado, en cuanto a la capacidad de identificación de Patrones de WF, es muy difícil comparar los algoritmos entre si ya que suelen utilizar estructuras de representación muy diferentes y difícilmente comparables de una forma automática. Aun así, en esencia, un experto humano podría identificar fácilmente que patrones han sido identificados mirando el WF inferido. Esta medida puede resultar un buen método de validación de la capacidad de identificación de los algoritmos de WM.

Por otro lado, la legibilidad es una característica muy subjetiva y difícil de medir. Modelos de representación que son perfectamente legibles para un grupo de expertos puede ser totalmente ilegible para otro. Dejando de lado este inconveniente, los modelos cuya talla es mayor (por ejemplo, mayor numero de estados y arcos) tienen una legibilidad menor. Estas características son fácilmente medibles, y pueden resultar una buena herramienta para medir la legibilidad del problema.

Una vez definida una métrica de evaluación, el siguiente paso es la definición de un método de experimentación que nos permita poner en práctica esta. En la realidad, los algoritmos de WM tienen utilizan WLogs con instancias que hayan visto modificadas su descripción conforme a cuando fueron diseñadas. Los algoritmos de WM en estos casos ayudan a descubrir patrones que se producen en la realidad y que no están siendo consi-

deradas en el diseño original. Sin embargo, este tipo de WLogs no pueden ser utilizados para probar la capacidad de inferencia de los modelos, simplemente porque se desconoce el modelo real. Por ello es necesario definir una metodología de experimentación basada en el conocimiento previo del modelo real para poder calcular la desviación producida en el modelo inferido.

Para ello se van a definir una serie de experimentos basados en la definición previa de WF que serán simulados para crear WLogs que puedan ser utilizados como corpus de entrada para experimentar con los algoritmos de WM. En esta linea vamos a realizar tres experimentos principales:

- El primer experimento, se basa en un corpus disponible en la literatura que contiene un conjunto de pequeños conjuntos de muestras que describen patrones comunes dentro de la disciplina de representación de WF. Con este corpus se va a medir la capacidad de inferencia de los modelos frente al marco general de la representación de WF.

- El segundo experimento, es un corpus basado en un problema real de Vías Clínicas que condensa los patrones de WF más comunes en el diseño de Vías Clínicas. Con este corpus se va a medir la capacidad de inferencia de los patrones más usuales en Vías Clínicas.

- El tercer experimento, también basado en Vías clínicas, esta pensado para evaluar patrones de WF muy complejos como rutinas paralelas entrelazadas, que solucionarían problemas como la identificación de secciones críticas en los WF. Con este corpus evaluaremos la capacidad de inferencia de patrones complejos con secuencias paralelas largas por parte de los algoritmos de inferencia.

Utilizando estos experimentos se va a comparar PALIA con otros algoritmos ya existentes en la literatura. De entre los algoritmos existentes se han seleccionado cuatro algoritmos, que son: el Heuristic Miner(HM) [109], el Genetic Process Miner(GPM) [41], el α [104] y el $\alpha++$ [40]. Estos algoritmos han sido escogidos porque tienen una implementación disponible. Estos algoritmos se dividen en dos tipos: los capaces de inferir Redes de Petri como α y $\alpha++$ y los que son capaces e inferir DFA como HM y GPM. Los algoritmos α y $\alpha++$ supondrán un buen método para medir la capacidad de inferencia de WF. Por otro lado los algoritmos HM y GPM supondrán un buen test para comprobar como se comportan los modelos no paralelos con los experimentos propuestos. Además, nos van a servir de medida para comprobar la legibilidad del modelo, ya que estos algoritmos infieren modelos muy legibles, basados en autómatas simples de una complejidad gramatical equivalente a los TPA.

9.1.1. Métricas de evaluación de los Algoritmos de Minería de Flujos de Trabajo

Los algoritmos presentados en la literatura no han abordado el tema de la evaluación de los algoritmos en comparación con otros algoritmos, trasladando al usuario la responsabilidad de decidir si el WF inferido es correcto o no. En [33] hubo un intento de formalizar una métrica para la evaluación de algoritmos de WM, por un lado mediante validación subjetiva de los eventos identificados y por otro utilizando un algoritmo de comparación

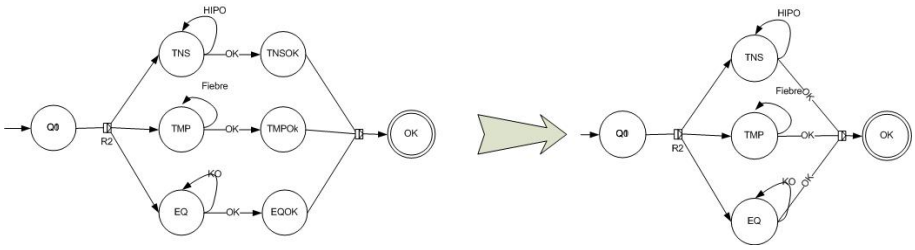

Figura 9.1: Ejemplo de dos Workflows equivalentes con diferente eficiencia

de WF mediante medición de distancias de edición. Sin embargo, el algoritmo de distancias tiene una complejidad muy alta por lo que otros algoritmos más modernos como el Genetic Process Miner [41] o el Frequency Abstraction Miner [84] utilizan medidas de distancia basadas en la utilización de evaluación mediante corpus de test. Sin embargo, todos estos trabajos se dedican a validar únicamente la eficiencia del modelo inferido con respecto al modelo original, o como mucho validando la eficiencia del algoritmo frente a la incorporación de ruido en el corpus de entrada, no existiendo trabajos que comparen el acierto de otros algoritmos frente a los mismos corpus.

Además, la utilización de técnicas de distancia de edición no siempre son adecuadas para validar sistemas de WM. Al comparar un WF resultante, con el originalmente ejecutado, si dos patrones son equivalentes, y el inferido utiliza menos transiciones, es penalizado en el resultado cuando realmente el algoritmo inferido se corresponde con una mejora sobre el WF original. En la Figura 9.1 se puede ver un ejemplo de como un WF inferido puede ser más eficiente que el WF original. En la parte de la izquierda se puede observar un WF diseñado manualmente que realiza tres actividades, y que cuando tienen resultados erróneos se repiten, y cuando tienen resultados correctos pasan a un estado de espera. El esquema de la derecha es un WF inferido a partir de los WLogs generados del primero, que es equivalente en su funcionamiento pero es más eficiente porque requiere menos nodos y arcos que el original. Este ejemplo hubiese sido penalizado por el método de validación por distancia de edición.

Por otro lado, otra forma de evaluar los modelos sería la creación de corpus de Train-Test para evaluar el acierto del modelo. Sin embargo, esta técnica no resulta adecuada para modelos que no contengan muestras de test reales. Además, muchos de los algoritmos de WM estudiados están más pensados para proponer una visión gráfica del problema, que para proponer una solución real del flujo de trabajo, por lo que realizan generalizaciones que ni siquiera admitirían ninguna de las muestras de entrada. No obstante, estos algoritmos priman la identificación de patrones de WF por encima de la identificación de instancias de WF. De este modo, aunque para PALIA sería un buen método de validación, el resto de los algoritmos obtendrían resultados muy pobres. Esto resultaría una fortaleza injusta de PALIA con respecto a sus competidores que supondría un sesgo.

9.1.2. Evaluación Propuesta

Basándose en las premisas anteriores, para evaluar PALIA con respecto a otros algoritmos, se proponen unas métricas de evaluación. Estas métricas van a tener como objetivo primordial evaluar los dos puntos más importantes en el diseño de WF, la eficacia y la legibilidad:

- La *Evaluación del acierto del modelo(Eficacia).*- El objetivo es valorar la capacidad que tiene el algoritmo por encontrar el modelo real que ha generado el corpus. Para ello se va a optar por una evaluación subjetiva. De cada uno de los corpus se van a listar los Patrones de WF que contiene el WF original, y un experto identificará los patrones del proceso original en el WF inferido. Así, para cada uno de los experimentos se definirá una planilla con los patrones existentes en el WF original. En esta planilla un experto humano, observando los resultados de los algoritmos, marca aquellos patrones que han sido correctamente identificados. A esta planilla se le denomina *Planilla de Eficacia.* De este modo se consigue una medición de la capacidad de inferencia de los algoritmos mediante el ratio entre el número de patrones totales y el número de patrones identificados. Al ratio entre el numero de patrones identificados por el algoritmo y el número de patrones total se le denominara *Eficacia.*

$$Eficacia = \frac{Patrones_{Identificados}}{Patrones_{Totales}}$$

- *Evaluación de la legibilidad.*- El objetivo de este punto es evaluar la legibilidad que tiene el sistema. Para ello se va a proponer una medida basada en la talla de las transiciones. Por ejemplo, el número de nodos y arcos con la que es representado un WF es indirectamente proporcional a la facilidad con la que se puede leer. En esta línea, coeficientes que pueden aportar información sobre la validación del WF inferido en cuanto a la legibilidad se refiere son los Nodos por Arco (Cf_{NxA}) y el Coeficiente de Ramificación (Cf_{Rm}).

  El Cf_{NxA} es el producto del numero de nodos por el numero de arcos. Este coeficiente puede dar una visión global de la legibilidad del WF inferido. Cuanto mayor sea este coeficiente, más comprometida se encontrará la legibilidad, ya que el WF será más grande.

  Otro coeficiente que puede ayudar a medir la legibilidad de los WF es El Cf_{Rm}. El Cf_{Rm} mide la ramificación media que se está produciendo en el WF inferido. La ramificación es una medida basada en la cardinalidad del rango y el dominio de las transiciones. Para las transiciones cuyo dominio sea uno el Cf_{Rm} será igual a la cardinalidad del rango de la transición. Para las transiciones cuyo dominio sea mayor que uno su Cf_{Rm} sera equivalente a la multiplicación de la cardinalidad del dominio por la cardinalidad del rango de la transición. El Cf_{Rm} de un WF es la media de los coeficientes de todas sus transiciones. Sin embargo, este coeficiente resulta muy dependiente del sistema de representación, y puede resultar un importante sesgo a la hora de comparar modelos diferentes.

  Por otro lado, si incorporamos información al Cf_{NxA} sobre los patrones inferidos podremos calcular como de legible es el modelo con respecto a los patrones que ha

sido capaz de identificar.

Para ello calculamos el *Coeficiente de Legibilidad Relativa* del modelo(Cf_{LR}) del siguiente modo

$$Cf_{LR} = \frac{\frac{Cf_{NxA_{Inferido}}}{Patrones_{Identificados}}}{\frac{Cf_{NxA_{Real}}}{Patrones_{Real}}}$$

Donde $Cf_{NxA_{Inferido}}$ es el coeficiente Cf_{NxA} del WF inferido por el algoritmo, $Patrones_{Identificados}$ es el numero de patrones correctamente identificados por el algoritmo, $Cf_{NxA_{Real}}$ es el coeficiente Cf_{NxA} del WF real y $Patrones_{Real}$ es el numero de patrones existentes en el WF real.

El Cf_{LR} es una medida de la talla del WF inferidos con respecto a los patrones identificados. Esta medida es como el ratio entre el numero de estructuras inferidas ($Cf_{NxA_{Inferido}}$) por patrón identificado y el número de estructuras reales ($Cf_{NxA_{Real}}$) por patrón real. Intuitivamente, cuando el número de patrones identificados es pequeño, o el numero de estructuras reales es muy grande el coeficiente tiende a infinito, por lo que la legibilidad será mala. Por otro lado cuando el numero de patrones identificados y el número de estructuras inferidas tiende a ser el iguales. El valor del ratio tiende a ser igual a uno. Por último, si el numero de estructuras utilizadas por patrón identificado, es menor que el WF real, el valor tenderá a cero, por lo que la legibilidad será mejor que el WF original.

En muchas ocasiones, el algoritmo utilizado para aprender el modelo tiene como herramienta de representación un lenguaje diferente al de el WF original. Esto puede suponer un sesgo en la evaluación del sistema. La medida de Legibilidad Relativa está siempre asociada a la legibilidad del WF original, pero no supone que la legibilidad del WF original es la mejor. Si esta medida es menor de 1 significará que la legibilidad del WF inferido con respecto a los patrones identificados es mejor que la del WF original. Por esto, esta medida puede ayudar a atenuar los efectos del posible sesgo causado.

9.1.3. Algoritmos de aprendizaje de Flujos de Trabajo usados

Para comparar los resultados de eficacia y la legibilidad relativa del los resultados del algoritmo presentado se van utilizar cuatro algoritmos de WM conocidos en la literatura usando la implementación que existe de estos algoritmos en el paquete ProM [108] del que ya se habló anteriormente. Estos algoritmos son: el Heuristic Miner(HM) [109], el Genetic Process Miner(GPM) [41], el α [104] y el $\alpha++$ [40].

En la Tabla 9.1 se hace un cuadro resumen de las diferencias entre los algoritmos utilizados. Debido a las grandes diferencias entre los algoritmos, antes de comenzar la fase de experimentación hay que tener en cuenta las siguientes consideraciones:

- Los algoritmos HM, GPM, α y $\alpha++$ son algoritmos basados en el paradigma de EBWM, mientras que PALIA es un algoritmo basado en el paradigma ABWM. La diferencia entre paradigmas está sobre todo en el etiquetado de las transiciones con

	PALIA	HM	GPM	α	$\alpha++$
Aprendizaje Secuencias Paralelas	Si	No	No	Si	Si
Aprendizaje de Actividades	Si	No	No	No	No
Modelo Resultado	TPA	DFA	DFA	RP	RP

Tabla 9.1: Características de los Algoritmos Estudiados

los resultados de las acciones. Sin embargo, las medidas de eficacia y legibilidad propuestas, únicamente van a valorar a la capacidad de inferencia de los flujos, independientemente del etiquetado de las transiciones. El modelo de evaluación propuesto, por lo tanto, es independiente del paradigma de WM utilizado.

- Los algoritmos HM, GPM, α y $\alpha++$ pueden recibir como entrada los eventos de tres formas diferentes: corpus con únicamente los eventos de Inicio (Corpus I), corpus con únicamente los eventos de Fin (Corpus F), o corpus con tanto eventos de Inicio como de Fin (Corpus IF). En los Corpus IF, todos estos algoritmos, utilizan los eventos de Inicio y Fin como si se tratasen de acciones diferentes por lo que los WF resultantes tiene una talla mayor y por lo tanto una legibilidad menor. Por lo tanto, ante resultados de eficacia equivalentes, los corpus I y F tiene mejores resultados que los corpus IF. PALIA utiliza corpus IF, pero trata estos eventos como una única actividad, por lo que su legibilidad no se encuentra comprometida. Esto va ha hacer que el método experimental para los algoritmos diferentes de PALIA vayan a utilizar 3 configuraciones diferentes de corpus que lo único que hacen es limitar la información contenida en pos de mejorar la legibilidad.

- Los algoritmos PALIA, α y $\alpha++$, son capaces de inferir flujos de trabajo paralelos por lo que son potencialmente capaces de aprender todos los modelos expuestos, por su parte, los algoritmos HM y GPM solo aprenden modelos basados en grafos que no admiten acciones paralelas. En este caso, los resultados obtenidos con los algoritmos modelos no paralelos nos van a permitir decidir en que casos es aceptable la utilización de estos modelos. Además los modelos no paralelos suelen ser más fáciles de aprender por lo que es posible que obtengan mejores resultados que algoritmos más complejos intentando aprender todo el modelo.

Para evaluar los algoritmos utilizando las consideraciones previas y las métricas propuestas se van a definir una serie de experimentos que van a comparar utilizando diferentes corpus. Para planificar estos experimentos de va a definir a continuación un método experimental que va a marcar el proceso de evaluación.

9.1.4. Método Experimental

Para evaluar los algoritmo es necesario encontrar corpus de aprendizaje de los cuales conozcamos el modelo original que nos permita comparar el modelo inferido con el modelo

real. En la literatura existen muy pocos corpus de WM disponibles que sirvan para evaluar este tipo de sistemas. Por otro lado en este trabajo se pretende evaluar la capacidad de los sistemas de WM para la inferencia de Vías Clínicas. Por ello es necesario crear corpus que contengan estructuras clásicas de la definición de Vías Clínicas.

El método de experimentación comienza seleccionado unos corpus de entrenamiento adecuados. De cada uno de corpus se destacan los patrones de WF que incorpore. Cada uno de los corpus se utiliza para aprender WF utilizando los algoritmos ya mencionados. A continuación un experto humano analizará el WF para detectar los patrones que haya identificado el WF inferido existentes en el WF original, ofreciendo así un valor de la eficacia del modelo. De hecho esta es la situación real que se encontraría un experto en procesos que quisiera utilizar herramientas basadas en WF, ya que, tendría de analizar el modelo inferido y detectar patrones de comportamiento. En estos experimentos de laboratorio, el experto juega con la ventaja de que conoce el WF original, dándole cierta ventaja a la hora de evaluar correctamente el acierto del algoritmo. Por otro lado, también se evaluará la legibilidad comparando la talla las estructuras inferidas con los patrones identificados.

Dicho esto, se pretenden realizar los siguientes tres experimentos principales:

- **Experimento Emit-Staffware**.- Este es un experimento disponible en la literatura y creado en la Universidad de Eindhoven para validar sistemas de WF. Este experimento se compone de 14 corpus, de los cuales,cada uno de ellos identifica un único Patrón de WF. En este caso la eficacia del modelo se medirá por el número de corpus de los que el algoritmo es capaz de identificar su patrón. En este caso dado que solo hay un patrón de identificar por corpus, no tiene sentido evaluar la legibilidad.

- **Experimento Carepaths**.- Este experimento esta formado por dos corpus basados en un modelo de Vías Clínicas. En este caso, los corpus se han conseguido mediante el diseño de un TPA y su posterior simulación vía TPAEngine en modo canónico para generar todas las muestras posibles, limitando el número de acciones iguales consecutivas para que la simulación no fuera infinita. En este caso se evaluará la efectividad y la legibilidad de cada uno de los algoritmos.

- **Experimento IC**.- Este experimento se corresponde con dos corpus basados en un modelo de Vías Clínicas que aborda unos patrones diferentes a los abordados por el experimento anterior. Los corpus de IC se han conseguido del mismo modo que el Experimento Carepaths, mediante la simulación utilizando el TPAEngine. En este caso también se evaluará la efectividad y la legibilidad de cada uno de los algoritmos.

A continuación se va a proceder a explicar con detalle cada uno de los experimentos y comentar los resultados obtenidos en los experimentos.

9.2. Experimento EMiT-Staffware

El experimento Emit-Staffware está basado en un corpus disponible [45] pensado para validar la eficacia del algoritmo de WM EMiT [46]. Este corpus consta de 91 muestras y está subdividido en 15 subcorpus pequeños. Cada uno de estos subcorpus ha sido creado

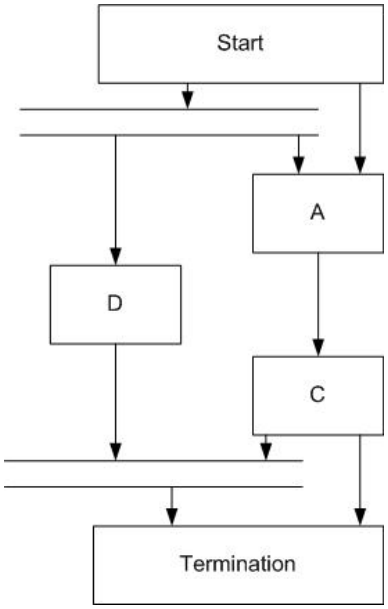

Figura 9.2: Workflow original del Corpus SW04

utilizando una muestra completa[1] de un WF diseñado y ejecutado en el motor de WF Staffware. Cada uno de estos subcorpus representa un patron específico, y está pensado para probar la capacidad de inferencia de los algoritmos para cada uno de estos patrones.

Los distintos corpus definen los siguientes patrones:

- El corpus *SW01* representa la ejecución de una acción simple en una secuencia. Este WF representa la expresión regular A.
- El corpus *SW02* representa un patrón de elección exclusiva por el que tras una acción A se produce una acción B o una acción C. Este WF representa la expresión regular $A(B|C)$.
- El corpus *SW03* representa la ejecución de un acción eventual. Tras la ejecución de una acción B puede ejecutarse una acción C o no, antes de ejecutarse la acción D. Este WF representa la expresión regular $(BD|BCD)$.
- El corpus *SW04* representa la ejecución de una secuencia de acciones que puede ejecutarse o no en paralelo con otra acción. Este WF se presenta en la Figura 9.2.
- El corpus *SW05* representa la ejecución de un ciclo simple entre tres acciones que se realiza al menos una vez. este A. Este WF representa la expresión regular $A(BCD)^{+}E$.
- El corpus *SW06* representa la ejecución paralela de dos acciones A e I que eventualmente se pueden ejecutar de forma paralela con la acción C. Este WF se presenta en la Figura 9.3.

[1]Una muestra completa es un conjunto de muestras que entre todas contienen todos los caminos posibles que puede seguir el WF

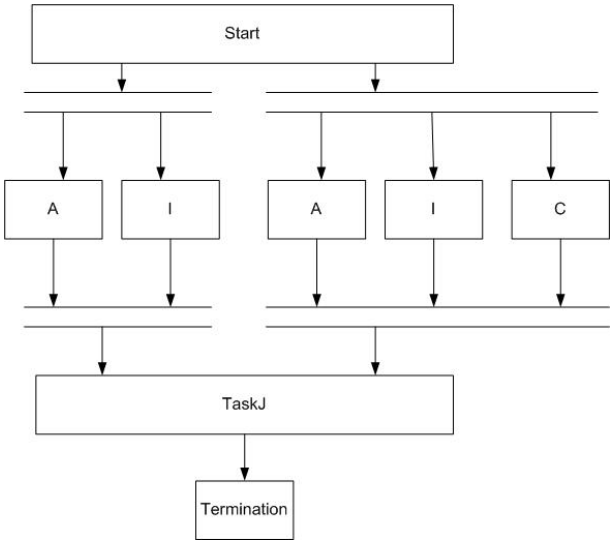

Figura 9.3: Workflow original del Corpus SW06

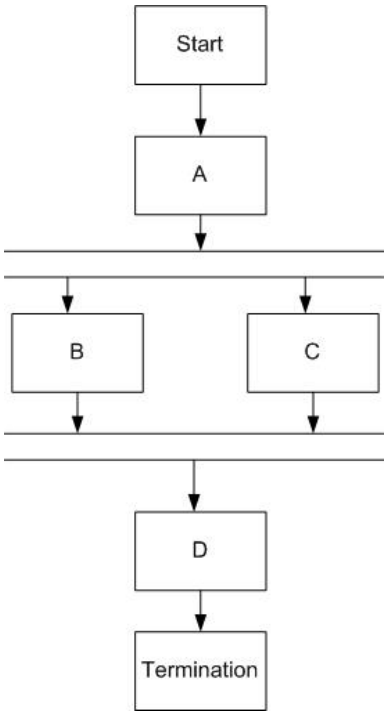

Figura 9.4: Workflow original del Corpus SW07

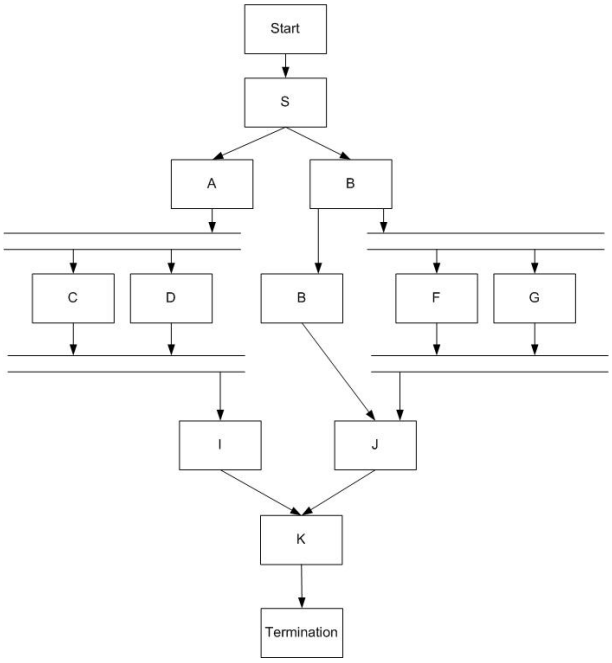

Figura 9.5: Workflow original del Corpus SW08

- El corpus *SW07* representa la separación y sincronización de dos acciones paralelas. Este WF se presenta en la Figura 9.4.
- El corpus *SW08* representa un WF completo con varias separaciones y sincronizaciones paralelas. Dos acciones paralelas que siempre se realiza, y otras dos acciones paralelas que sen pueden sustituir eventualmente con una acción simple. Este WF se presenta en la Figura 9.5.
- El corpus *SW09* representa un ciclo que involucra a tres acciones en el que el final del ciclo esta en medio. Este WF representa la expresión regular $A(BCD)^*BCE$.
- El corpus *SW10* representa un ciclo corto entre dos acciones. Este WF representa la expresión regular $A(BA)^*C$.
- El corpus *SW11* representa dos patrones paralelos de 3 acciones cada uno que comparten dos acciones, y que se sincronizan en una acción paralela. Este WF se presenta en la Figura 9.6.
- El corpus *SW12* representa una acción cíclica. Este WF representa la expresión regular A^+.
- El corpus *SW13* representa dos patrones paralelos que se sincronizan en una única acción. Este WF se presenta en la Figura 9.7.
- El corpus *SW14* representa dos patrones paralelos independientes, que comparten una acción. Este WF se presenta en la Figura 9.8.

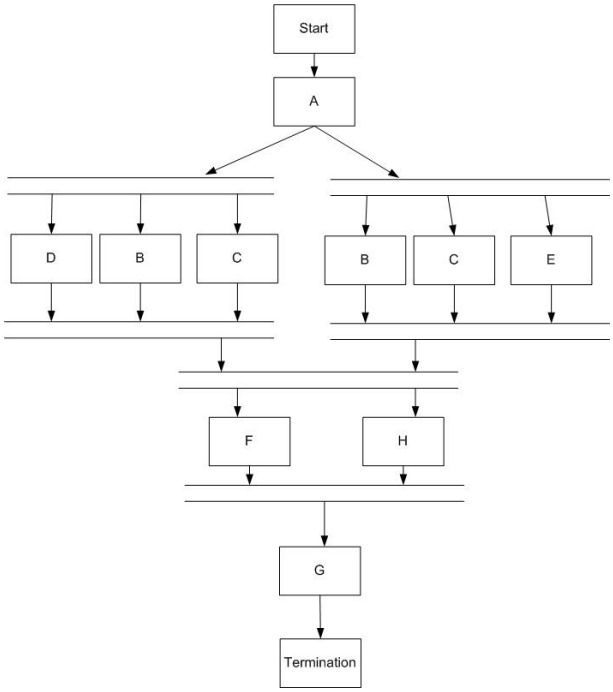

Figura 9.6: Workflow original del Corpus SW11

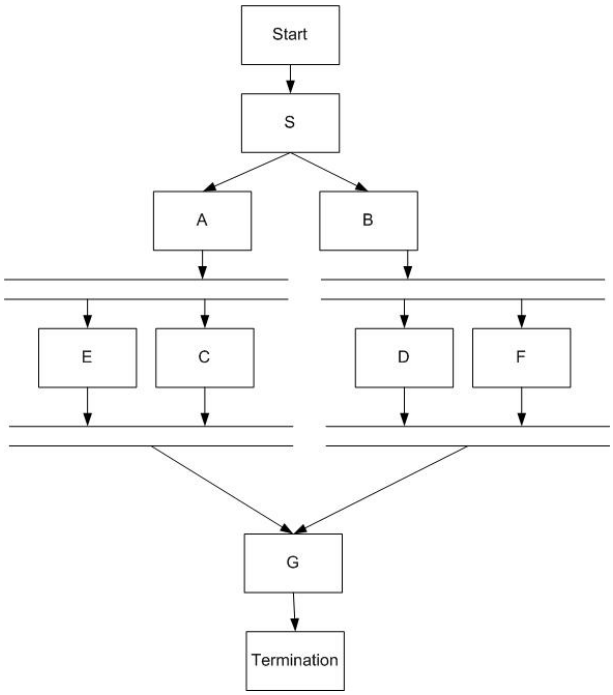

Figura 9.7: Workflow original del Corpus SW13

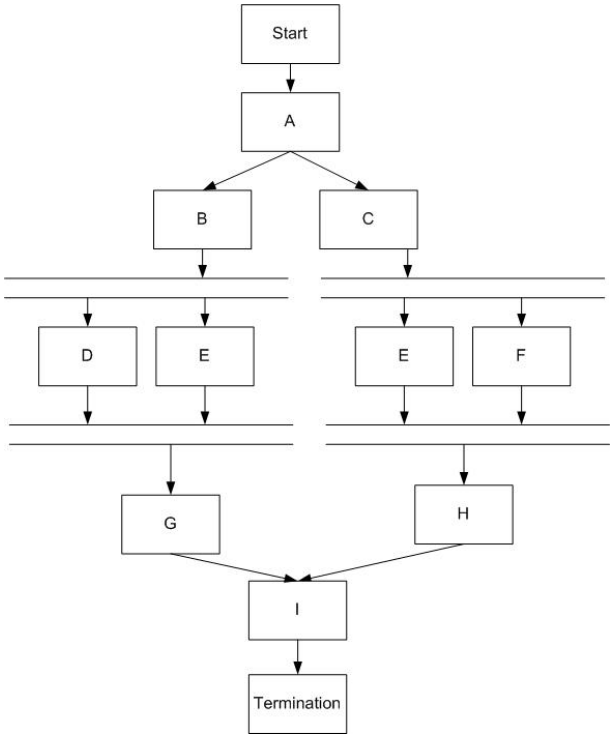

Figura 9.8: Workflow original del Corpus SW14

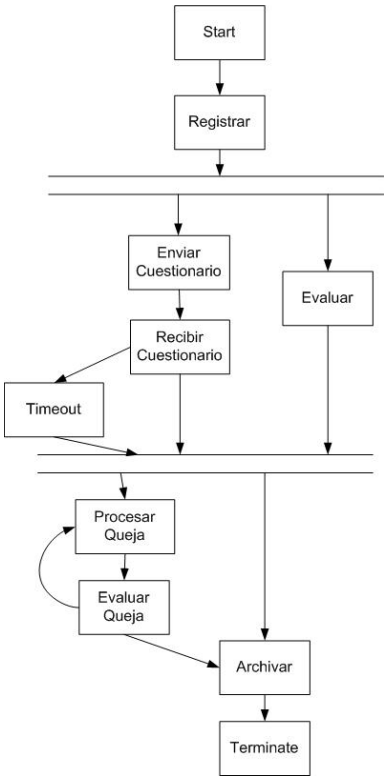

Figura 9.9: Workflow original del Corpus SW15

	PALIA	Heuristics Miner	Genetic Process Mining	Alpha Algorithm	Alpha++ Algorithm
SW01: Tarea Simple	X	X	X	X	X
SW02: XOR Split y XOR Join	X	X	X	X	X
SW03: Acción Opcional	X	-	X	X	X
SW04: XOR Paralelo o Simple	X	-	-	-	-
SW05: Ciclo Completo	X	X	X	-	X
SW06: Acción opcional paralela	X	-	-	X	X
SW07: AND Split y AND join	X	-	-	-	-
SW08: WF Completo Split y Joins	X	-	-	-	-
SW09: Ciclo Incompleto	X	X	X	X	X
SW10: Ciclo Corto	X	X	X	X	X
SW11: Join Paralelo Complejo	-	-	-	-	-
SW12: Autociclo	X	X	X	X	X
SW13: patrones paralelos sincronizados	X	-	-	-	-
SW14: patrones paralelos con acción compartida	X	-	-	-	-
SW15: Proceso Burocrático	X	-	-	-	-

Tabla 9.2: Planilla de eficacia del Experimento EMiT-Staffware

- El corpus *SW15* representa un proceso burocrático complejo de recepción de quejas. Este WF se presenta en la Figura 9.9.

En la Tabla 9.2 se muestra los resultados del Experimento Emit-Staffware. En esta Tabla, se han marcado los patrones de WF identificados por los distintos algoritmos. En esta Tabla se ha valorado únicamente la eficacia de los modelos. Esto es debido a que como los corpus son diferentes, no tiene sentido evaluar una legibilidad conjunta. De este experimento podemos sacar las siguientes conclusiones:

- El algoritmo PALIA tiene el mejor porcentaje de Eficacia (14/15) de todos los algoritmos teniendo únicamente problemas para definir los patrones paralelos que se sincronizan en otro patrón paralelo. De este modo podemos decir que PALIA se ha mostrado muy eficaz en la inferencia de WF generalmente usados.

- Los algoritmos α y $\alpha++$ han funcionado de forma parecida con excepción de los problemas deα los patrones de ciclos. En general los algoritmos α y $\alpha++$ tienen problemas para identificar correctamente las separaciones y sincronizaciones paralelas. Esto puede ser debido a tratar los eventos como acciones diferentes.

- A pesar de que los algoritmos HM y GPM tienen una baja eficacia en este corpus, es principalmente debido a que no son capaces de inferir los patrones paralelos. Es importante notar, que estos algoritmos están en un orden de acierto equivalente a los algoritmos α y $\alpha++$. Sin embargo, esto ha sido debido a que estos algoritmos no han identificado correctamente la mayoría de las separaciones y sincronizaciones paralelas y han identificado prácticamente todos los problemas no paralelos. El algoritmo GPM es capaz de identificar correctamente todos los patrones no paralelos (al igual que PALIA y $\alpha++$). Por su parte HM ha funcionado peor, ya que no ha sido capaz de inferir el WF de Acción Opcional (SW03).

9.2.1. Experimento CarePaths

Este experimento está basado en una Vía Clínica del cuidado de la Insuficiencia Cardíaca que fue creada en el proyecto europeo CAREPATHS [6]. Este experimento se ha creado específicamente para validar la capacidad de inferencia de los algoritmos de WM en los patrones más usuales en las Vías Clínicas. Para ello se han condensado estos patrones en un ejemplo de Vía Clínica simple. La intención de este flujo de procesos es probar todos los patrones existentes en las Vias Clínicas diseñadas en el citado proyecto

Este experimento está basado en dos subexperimentos complementarios. El primer experimento utiliza secuencias paralelas simples, es decir, que las secuencias paralelas tienen como máximo un elemento por rama. El segundo experimento es equivalente al primero pero utiliza secuencias paralelas de más de un elemento. Esto nos va a permitir evaluar como se comportan los algoritmos de inferencia ante el alargamiento de las secuencias paralelas. El problema de la detección de las secuencias paralelas es un problema clave, ya que define la capacidad de los algoritmos para detectar los grupos de acciones que se ejecutan secuencialmente de entre las acciones que se ejecutan paralelamente en un WF.

Los experimentos son los siguientes:

9.2.2. Experimento $CarePaths_A$

El Experimento $CarePaths_A$ trata de probar los algoritmos en condiciones de patrones simples de WF con condiciones de paralelismo simple. La Figura 9.10 muestra gráficamente el TPA del Experimento.

Este WF comienza por un cuestionario (Q_0) para el paciente y que decide si este debe seguir en la vía o por el contrario finalizarla si el paciente no es compatible con el tratamiento incluido en la Vía. Tras este primer filtro, el paciente pasa por un segundo cuestionario (Q_1) que decide si se le realiza un tratamiento normal o un tratamiento exhaustivo.

En caso de tratarse de un tratamiento normal, existen dos tipos de tratamientos que son seleccionados según un tercer cuestionario (Q_2). En ambos casos, el tratamiento se repite hasta que dé resultado.

En caso de tratarse de un tratamiento exhaustivo se realizan 3 actividades en paralelo: medir la tension (TNS), la temperatura (TMP) y el equilibrio hídrico (EQ). Cuando alguna de estas actividades falla, se repite hasta que sea correcta. cuando las tres actividades son correctas, se produce una sincronización de las actividades, junto con las secuencias de tratamiento normal, para pasar al cuestionario (Q_3) que valida si los tratamientos han

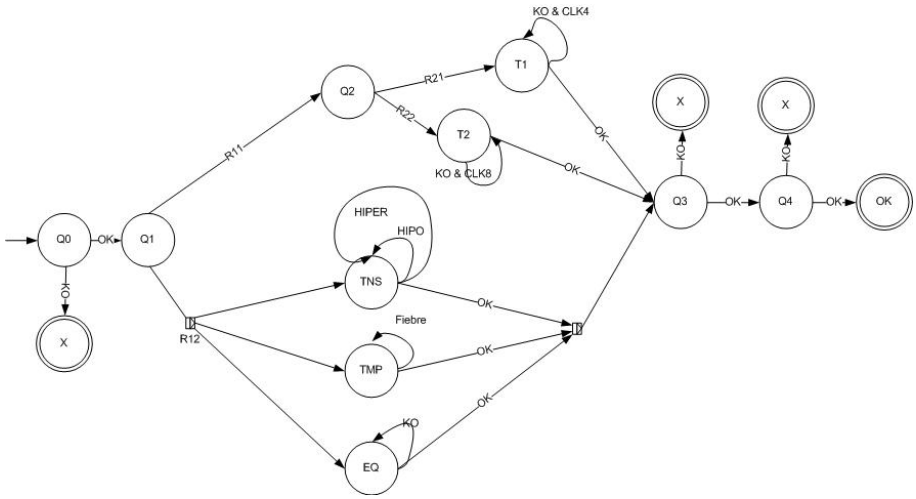

Figura 9.10: Estructura del Experimento $CarePaths_A$

tenido efecto, en cuyo caso pasaríamos al siguiente cuestionario (Q_4) que valida la calidad del servicio. Una vez pasado este último cuestionario se finalizaría la vía.

El experimento A ha sido realizado utilizando un corpus creado a partir de la simulación del TPA definido mediante TPA Engine, como resultado se han obtenido 318 muestras que han sido utilizadas para inferir WF.

En la Tabla 9.3 se muestra la planilla de eficacia del experimento. Esta tabla ha sido construida por un experto que tras analizar los WF inferidos, anotando si estos fueron capaces de identificar los patrones seleccionados o no. Los patrones del WF del experimento $CarePaths_A$ que fueron seleccionados son:

- *Estructura inicial:* Este patrón se refiere al cuestionario inicial (Q_0) y a su paso si no es pasado convenientemente al estado de fin de la via.
- *Multielección inicial:* Este patrón se refiere al cuestionario Q_1 y al paso al cuestionario Q_2 y tratamiento exhaustivo.
- *Secuencia T1:* Este patrón se refiere al paso desde el cuestionario Q_2 al tratamiento T_1, su bucle y su sincronización al cuestionario Q_3.
- *Secuencia T2:* Este patrón se refiere al paso desde el cuestionario Q_2 al tratamiento T_2, su bucle y su sincronización al cuestionario Q_3.
- *Separación paralela inicial:* Este patrón se refiere a si se ha realizado una separación paralela entre el cuestionario Q_1 y el tratamiento exhaustivo *TNS*, *TMP* y *EQ*.
- *Bucle equilibrio:* Este patron se refiere al bucle en la actividad paralela *EQ*.
- *Bucle tensión:* Este patron se refiere al bucle en la actividad paralela *TNS*.
- *Bucle temperatura:* Este patron se refiere al bucle en la actividad paralela *TMP*.

	PALIA	HM			GPM			Alpha			Alpha++		
		I/F	I	F	I/F	I	F	I/F	I	F	I/F	I	F
Estructura Inicial	X	X	X	X	X	X	-	X	X	X	-	X	X
Multielección Inicial	X	X	X	X	X	X	X	X	-	-	-	X	X
Secuencia T1	X	-	-	-	X	-	X	X	-	-	-	-	-
Secuencia T2	X	-	-	-	X	X	X	X	-	-	-	-	-
Separación Paralela Inicial	X	-	-	-	-	-	-	-	-	-	-	-	-
Bucle Equilibrio	X	-	-	-	-	X	-	-	-	-	-	-	-
Bucle Tension	X	X	X	X	X	X	X	X	-	-	-	-	X
Bucle Temperatura	X	X	X	X	X	X	X	-	-	-	-	X	-
Sincronización Paralela	X	-	-	-	-	-	-	-	-	-	-	-	-
Sincronización Exclusiva	X	X	X	X	X	-	X	X	-	-	X	-	X
Estructura Final	X	X	X	-	X	X	-	X	X	-	X	-	-

Tabla 9.3: Planilla de eficacia del Experimento $CarePaths_A$

		Nodos	*Arcos*	Cf_{NxA}	*Patrones*	Cf_{Rm}	*Eficacia*	Cf_{LR}
Real		12	19	228	11	1,92	-	-
PALIA		12	19	228	11	1,92	1	1
Heuristic Miner	I/F	21	32	672	6	1,52	0,55	5,40
	I	11	17	187	6	1,55	0,55	1,50
	F	10	15	150	5	1,5	0,45	1,45
Genetic Process Mining	I/F	21	35	735	8	1,67	0,73	4,43
	I	11	21	231	7	1,91	0,64	1,59
	F	10	17	170	6	1,7	0,55	1,37
α	I/F	21	34	714	7	2,62	0,64	4,92
	I	11	7	77	2	0,73	0,18	1,86
	F	10	6	60	1	0,4	0,09	2,89
α ++	I/F	21	30	630	2	7,38	0,18	15,20
	I	11	8	88	3	2,09	0,27	1,42
	F	10	8	80	4	1,7	0,36	0,96

Tabla 9.4: Resultados del Experimento $CarePaths_A$

- *Sincronización paralela:* Este patrón se refiere a que si ha producido un arco de sincronización paralela entre *TNS*, *TMP* y *EQ* y el cuestionario Q_3.
- *Sincronización exclusiva:* Este patrón se refiere a si existen transiciones simples entre los tratamientos T1 y T2 con el cuestionario Q_3.
- *Estructura final:* Este patron engloba las transiciones entre los cuestionarios Q_3 y Q_4.

En la Tabla 9.4 se puede observar los resultados de los coeficientes calculados de los distintos algoritmos.

En este cuadro podemos ver que PALIA no solo identifica todos los patrones sino que es capaz de inferir el modelo original con su misma legibilidad. El WF inferido por PALIA se puede ver en la Figura B.1.

Por su parte el algoritmo Heuristic Miner, cuyo WF inferido se muestra en la Figura B.2 del anexo B, ha conseguido resultados mediocres con una eficacia del 0,55 y una legibilidad del 1,50. Además del handicap inicial que le supone a este algoritmo el hecho de que la estructura inferida no admita transiciones paralelas, el Heuristic Miner no ha conseguido

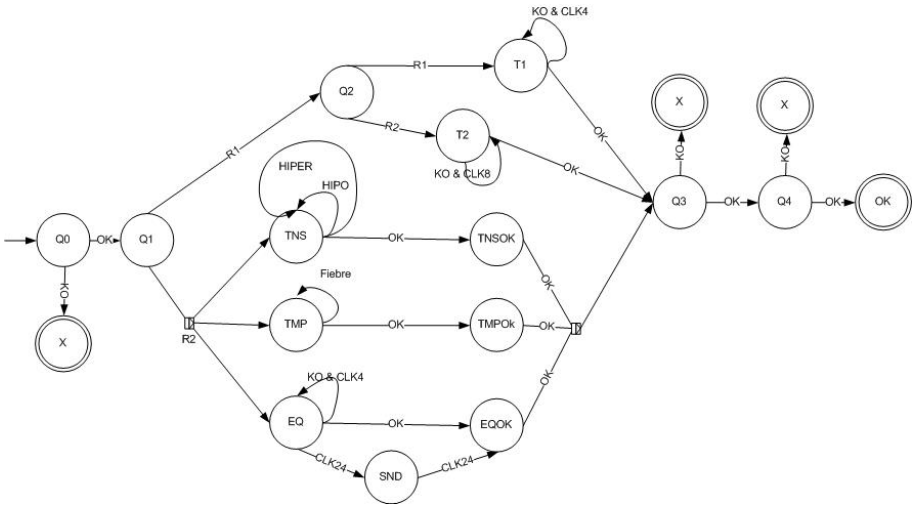

Figura 9.11: Estructura del Experimento $CarePaths_B$

identificar correctamente las secuencias simples ya que le ha faltado identificar los bucles de los tratamientos T_1 y T_2. En cuanto al tipo de corpus utilizados, el de datos iniciales y finales ha funcionado en efectividad igual que el de datos iniciales, pero ha sido descartado al ser mucho menos legible.

Por otro lado el algoritmo Genetic Process Mining ha tenido mejores resultados con la información inicia y final que el resto de algoritmos de EBWM con una eficacia del 0,73. La Figura B.3 del anexo B muestra el WF inferido. Este resultado es bastante aceptable teniendo en cuenta que este algoritmo cuenta con el handicap de que no es capaz de inferir secuencias paralelas. No obstante, su mayor marca la ha logrado con el corpus de información inicial y final, lo que sin duda ha influido en su baja legibilidad (4,43).

En la Figura B.4 del anexo B se muestra el WF inferido por el algoritmo α. Este algoritmo ha funcionado mucho mejor con el corpus IF que con el resto de corpus. El WF inferido a pesar de tener una eficacia aceptable (0,64) tiene una legibilidad sea baja (4,92) que se muestra patente en dicha Figura.

Por último el algoritmo α++ no ha conseguido mejorar los resultados del algoritmo α obteniendo una eficacia de 0,36. A pesar de que su legibilidad relativa es mas baja que el modelo original (0,96) esto es debido a que prácticamente ha eliminado gran parte de los arcos quedando un WF muy sencillo que no explica el modelo tal y como se presenta en la Figura B.5 del anexo B, aunque los patrones que identifica lo hace con cierta legibilidad.

9.2.3. Experimento $CarePaths_B$

El Experimento $CarePaths_B$, mostrado en la Figura 9.11, es una simple modificación del experimento A que complica su inferencia notablemente. La modificación realizada consiste en poner estados de espera tras las ejecuciones paralelas de las secuencias de control exhaustivo. Esta simple modificación exigirá a los algoritmos a que no solo sean capaces de detectar las separaciones y sincronizaciones paralelas, sino que sean capaces de construir las secuencias paralelas asignando a cada secuencia sus actividades correspondientes.

	PALIA	HM			GPM			Alpha			Alpha++		
		I/F	I	F	I/F	I	F	I/F	I	F	I/F	I	F
Estructura Inicial	X	X	X	X	X	X	-	X	-	X	-	-	X
Multielección Inicial	X	X	X	X	X	X	X	-	-	-	-	-	X
Secuencia T1	X	-	-	-	-	-	X	X	-	-	-	-	-
Secuencia T2	X	-	-	-	-	-	X	X	-	-	-	-	-
Separación Paralela Inicial	X	-	-	-	-	-	-	-	-	-	-	-	-
Bucle Equilibrio	X	-	-	-	-	-	-	-	-	-	-	X	-
Bucle Tension	X	-	-	X	-	X	X	X	-	-	-	X	X
Bucle Temperatura	X	-	X	X	X	X	X	X	-	-	-	X	X
Paso Equilibrio OK	X	-	-	-	-	X	X	X	X	X	-	X	X
Paso Tension OK	X	X	X	X	X	X	X	X	-	-	-	-	-
Paso Temperatura OK	X	X	X	X	-	X	X	X	-	-	-	-	-
Sincronización Paralela	X	-	-	-	-	-	-	-	-	-	-	-	-
Sincronización Exclusiva Final	X	X	X	X	-	-	X	X	-	-	-	-	X
Estructura Final	X	X	X	-	X	X	-	X	X	-	-	-	X

Tabla 9.5: Planilla de eficacia del Experimento $CarePaths_B$

Además de este cambio, se ha modificado uno de los patrones paralelos, el de equilibrio, añadiendo un patrón de elección exclusiva. En este caso la secuencia de equilibrio funcionaría de forma diferente al Experimento A. Una vez alcanzado la actividad *EQ*, que se ejecuta de forma paralela a *TNS* y *TMP*, si al ejecutarse el resultado es *OK* se pasaría al estado de espera *EQOK* donde realizarían una acción de confirmación del fin de la acción. Sin embargo en el caso que pasaran más de 24 horas, se ejecutaría la actividad de sondaje (*SND*) para luego tras 24 horas más pasar al estado de espera *EQOK*. Las demás actividades paralelas, *TNS* y *TMP*, únicamente añaden el estado de espera *TNSOK* y *TMPOK*, respectivamente, al que pasan en caso de que el resultado de las acciones sea el esperado.

El Experimento $CarePaths_B$ ha sido realizado utilizando un corpus creado a partir de la simulación del TPA definido mediante TPA Engine, como resultado se han obtenido 540 muestras que han sido utilizadas para inferir WF.

En la Tabla 9.5 se la planilla de eficacia donde se muestran los patrones seleccionados para su identificación en el Experimento $CarePaths_B$. Es de notar que son muy parecidos a los del experimento anterior con la excepción de que añade tres patrones más. Los tres patrones añadidos con respecto al experimento anterior son:

- *Paso EquilibrioOK*: Este patrón se refiere al paso inequívoco siguiendo una transición simple desde desde la actividad *EQ* tanto a la actividad *EQOK* como a la actividad *SND*. También engloba la transición entre la actividad *SND* y la actividad *EQOK*.

- *Paso TensionOK*: Este patrón se refiere al paso inequívoco siguiendo una transición simple desde desde la actividad *TNS* a la actividad *TNSOK*.

- *Paso TemperaturaOK*: Este patrón se refiere al paso inequívoco siguiendo una transición simple desde desde la actividad *TMP* a la actividad *TMPOK*

		Nodos	*Arcos*	Cf_{NxA}	*Patrones*	Cf_{Rm}	*Eficacia*	Cf_{LR}
Real		16	24	384	14	1,75	-	-
PALIA		16	33	544	14	2,44	1	1,38
Heuristic Miner	I/F	29	55	1479	6	1,9	0,43	9,69
	I	15	29	435	7	1,93	0,50	2,27
	F	14	22	308	7	1,57	0,50	1,60
Genetic Process Mining	I/F	29	61	2,1	5	319	0,36	12,90
	I	15	29	510	8	1,93	0,57	1,98
	F	14	27	406	9	1,93	0,64	1,53
α	I/F	29	67	1769	10	4,76	0,71	7,08
	I	15	15	240	2	1,2	0,14	4,10
	F	14	11	154	2	0,79	0,14	2,81
α ++	I/F	-	-	-	-	-	-	-
	I	15	23	345	4	5,47	0,29	3,14
	F	14	19	266	7	4,36	0,50	1,46

Tabla 9.6: Resultados del Experimento $CarePaths_B$

En la Tabla 9.6 se puede observar los resultados de los coeficientes calculados de los distintos algoritmos.

En la Figura B.6 del anexo B se puede ver el WF inferido por el algoritmo PALIA. Aunque el algoritmo ha sido capaz de identificar los patrones definidos, como era de esperar, ha encontrado más problemas para encontrar el modelo que en el experimento A. Esto ha repercutido en que PALIA haya creado más transiciones erróneas. lo que ha contribuido a que la legibilidad haya llegado al 1,38. Sin embargo tiene el mejor índice de legibilidad de todos los algoritmos

El algoritmo Heuristic Miner prácticamente ha mantenido su eficacia (0,5) en este experimento con respecto al experimento A. Como podemos ver en las Tablas, el Heuristic Miner ha tenido más problemas para reconocer las repeticiones de las actividades paralelas. Por otro lado, no ha tenido problemas con los estados de espera de las actividades de la toma de temperatura y tensión, aunque no ha sido capaz de reconocer el patron de las secuencias paralelas más complicado (*paso Equilibrio Ok*). En cuanto a la legibilidad del WF, esta ha empeorado muy ligeramente con respecto experimento anterior llegando al 1,60. El WF inferido por este algoritmo se muestra en la Figura B.7 del anexo B.

La Figura B.8 del anexo B muestra el mejor WF inferido con GPM para el Experimento $CarePaths_B$. El algoritmo ha empeorado sus resultados en cuanto a la eficacia (0,64), con respecto al experimento anterior. Sin embargo, la legibilidad de éste es notablemente mejor (1,53). Esto es debido a que el anterior experimento consiguió mejores resultados con el corpus IF y su legibilidad se vio notablemente afectada. Observando los resultados de legibilidad corpus a corpus en comparación con los del experimento A se puede observar que los del Experimento $CarePaths_B$ se han empeorado.

El algoritmo α, cuyo WF inferido con corpus IF se muestra en la Figura B.9 del anexo B ha mejorado sus resultados con este modelo (0,71), sin embargo, ha empeorado notablemente su legibilidad. Esto podría ser achacable a que se ha inferido con el corpus IF, sin embargo no lo es, ya que, comparando los resultados corpus a corpus con los del Experimento A, podemos ver como la legibilidad empeora mucho utilizando el mismo tipo de corpus.

El WF inferido para el Experimento $CarePaths_B$ por el algoritmo α++ se muestra en la Figura B.10 del anexo B. Este ha sido conseguido utilizando el corpus de tipo F. El algoritmo utilizado (Implementado en ProM) ha dado error con el corpus IF, probablemente por que el alto numero de arcos y estados, por lo que aunque hubiese terminado

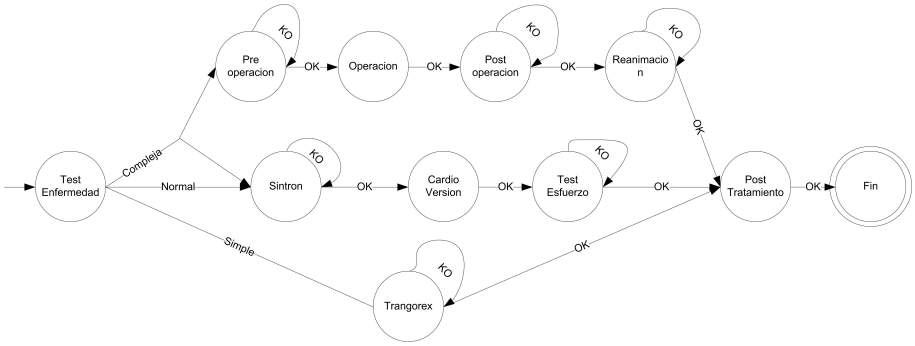

Figura 9.12: Estructura del Experimento IC_A

correctamente, sería demasiado alto haciendo ilegible el modelo. La eficacia del modelo ha mejorado notablemente desde el experimento anterior, aunque no llega a estar entre los mejores, sin embargo su legibilidad ha empeorado un poco.

9.3. Experimento IC

El Experimento IC_A corresponde a un modelo diferente a los experimentos A y B. Este experimento es una simplificación libre de un protocolo de actuación sacado de una Vía Clínica de Insuficiencia Cardíaca [52].

El objetivo de este WF es probar el comportamiento de los algoritmos ante un patrón complejo de WF para describir secciones críticas, como puede ser una Rutina Paralela Entrelazada (Ver Anexo A: Patrón 17) y sistemas de multielección (Ver Anexo A: Patrón 6) y multifusión (Ver Anexo A: Patrón 8) que comparten una rama de forma paralela. Este experimento se divide en dos subcorpus. El primero describe un proceso en el que se encuentra una rutina paralela con dos secuencias largas, que pueden ejecutarse al mismo tiempo. El segundo subcorpus añade una restricción al modelo por lo que las secuencias paralelas no pueden ejecutarse a la vez, ya que las secuencias han de ejecutarse de forma atómica.

Los experimentos son los siguientes:

9.3.1. Experimento IC_A

El TPA que define el proceso se muestra en la Figura 9.12. El flujo de procesos es el siguiente: tras un test inicial (*TestEnfermedad*) se decide el tipo de enfermedad.

Si el tipo de la enfermedad es simple se administra *Trangorex* hasta el fin del tratamiento. Si la enfermedad es normal se administra *Sintron* hasta que la sangre esté al nivel de coagulación deseado y entonces realizar una *Cardioversión*, que una vez realizada se realiza una *Prueba de Esfuerzo* para observar como ha funcionado la cardioversión.

En el caso de ser una enfermedad compleja se realiza la secuencia del *Simtron* paralelamente a la secuencia con una operación. Primero se realiza un *Preoperatorio* que se

	PALIA	HM			GPM			Alpha			Alpha++		
		I/F	I	F	I/F	I	F	I/F	I	F	I/F	I	F
Separación Inicial	X	-	-	-	-	X	-	X	-	-	-	X	X
Secuencia Simple	X	X	X	X	-	-	-	X	-	-	-	X	X
Bucle Trangorex	X	-	-	-	-	-	X	X	-	-	-	X	X
Secuencia Normal	X	-	-	-	-	-	X	X	-	-	-	X	X
Bucle Sintron	X	-	-	-	X	X	X	X	-	-	-	X	X
Bucle Prueba Esfuerzo	X	-	-	-	X	X	X	X	X	X	-	X	X
Separación Compleja	X	-	-	-	-	-	-	-	-	-	-	-	-
Secuencia Compleja	X	-	-	-	-	-	-	X	-	-	-	X	X
Bucle Posoperatorio	X	X	X	X	X	-	X	X	-	-	-	X	-
Bucle Reanimación	X	-	-	-	-	-	-	-	-	-	-	-	-
Secuencia Final	X	X	X	-	X	X	-	X	X	-	-	X	-

Tabla 9.7: Planilla de eficacia del Experimento IC_A

efectúa hasta que el paciente esté preparado, se realiza la *operación*, posteriormente se pasa al *Posoperatorio* y una vez finalizado y el paciente se pasa a *Reanimación*.

Sea el tipo de enfermedad que sea las secuencias se sincronizan realizando la actividad *Post Tratamiento* y posteriormente se finaliza la via.

El Experimento IC_A ha sido realizado utilizando un corpus creado a partir de la simulación del TPA definido mediante TPAEngine, como resultado se han obtenido 326 muestras que han sido utilizadas para inferir.

En la Tabla 9.7 se muestran una serie de patrones que pertenecen al WF del experimento, y si han sido correctamente identificados por los algoritmos utilizados. Esta tabla ha sido construida por un experto que tras analizar los WF inferidos anotó si estos fueron capaces de identificar los patrones seleccionados o no. Los patrones del WF seleccionados son:

- *Separación Inicial*: Este patrón se refiere a la actividad *TestEnfermedad* y las transiciones que pasen a los tres tipos diferentes de transiciones, simple, normal y compleja.
- *Secuencia Simple*: La secuencia simple se refiere a la secuencia que sale de la actividad *TestEnfermedad*, posteriormente se ejecuta la actividad *Trangorex* y seguidamente *PostTratamiento*.
- *Bucle Trangorex*: Este patrón se refiere a la transición en que une la actividad *Trangorex* consigo misma.
- *Secuencia Normal* Este patrón se refiere a a la secuencia de actividades *TestEnfermedad*, *Simtron*, *Cardioversion*, *PruebaEsfuerzo* y *PostTratamiento*.
- *Bucle Sintron*: Este patrón se refiere al bucle en la actividad *Sintron*.
- *Bucle Prueba Esfuerzo*: Este patrón se refiere al bucle en la actividad *PruebaEsfuerzo*.
- *Separación Compleja*: Este patrón se refiere a la separación que va desde la actividad *TestEnfermedad* a la actividad *Preoperacion* y *Sintron* de forma paralela.

		Nodos	*Arcos*	Cf_{NxA}	*Patrones*	Cf_{Rm}	*Eficacia*	Cf_{LR}
Real		11	18	198	11	1,67	-	-
PALIA		11	22	242	11	2,55	1	1,22
Heuristic Miner	I/F	22	47	1034	3	2,14	0,27	16,15
	I	12	28	336	3	2,33	0,27	6,22
	F	10	26	260	2	2,6	0,18	7,22
Genetic Process Mining	I/F	22	43	946	4	1,95	0,36	13,14
	I	12	24	288	4	2	0,36	4,00
	F	10	25	250	5	2,5	0,45	2,78
α	I/F	22	42	924	9	5,64	0,82	5,70
	I	12	9	108	2	1,42	0,18	3,00
	F	10	7	70	1	1,5	0,09	3,89
α ++	I/F	-	-	-	-	-	-	-
	I	12	13	156	9	3,08	0,82	0,96
	F	10	11	110	7	3,5	0,64	0,87

Tabla 9.8: Resultados del Experimento IC_A

- *Secuencia Compleja*: Este patrón se refiere a la secuencia de actividades *Preoperatorio*, *Operacion*, *Postoperacion*, *Reanimacion* y *PostTratamiento*.

- *Bucle Postoperatorio*: Este patrón se refiere al bucle de la actividad *Postoperatorio*.

- *Bucle Reanimación*: Este patrón se refiere al bucle de la actividad *Reanimacion*.

- *Secuencia Final*: Este patrón se refiere a la secuencia *PostTratamiento* a *Fin*.

En la Tabla 9.4 se puede observar los resultados de los coeficientes calculados de los distintos algoritmos.

Como podemos ver en la tabla la eficacia de PALIA sigue siendo alta, ya que es capaz de identificar todos los patrones. En cuanto a la legibilidad, PALIA consigue una legibilidad relativa de 1,22 bastante parecida a la original. En la Figura B.11 del anexo B se muestra el WF inferido por PALIA utilizando el corpus del Experimento IC_A.

El algoritmo Heuristic Miner tiene mayores problemas que en los experimentos anteriores para identificar los patrones. Esto queda patente en su baja eficacia (0,27) junto con una mala legibilidad relativa (6,22). El WF resultante de la inferencia del algoritmo se muestra en la Figura B.12 del anexo B.

Los resultados del algoritmo Genetic Process Mining viene a confirmar la baja eficacia de los algoritmos basados en modelos AFD, aunque mejor que el resultado del Heuristic Miner con un acierto máximo de 0,45. Además la legibilidad relativa da un resultado mejor que el Heuristic Miner (2,78). El WF inferido por el Algoritmo GPM se muestra en la Figura B.13 del anexo B.

El algoritmo α ha conseguido relativamente buenos resultados de eficacia (0,82) en este experimento aunque con una mala legibilidad relativa (5,70) debido a que los resultados buenos, han sido conseguidos con el corpus IF. El WF inferido por el algoritmo α se muestra en la Figura B.14 del anexo B.

EL algoritmo α++, al igual que en el Experimento $CarePaths_B$ solo ha funcionado con los corpus I y corpus F. Este algoritmo ha conseguido buenos resultados de eficacia (0,82) y legibilidad relativa (0.96). El WF inferido se muestra en la Figura B.15 del anexo B.

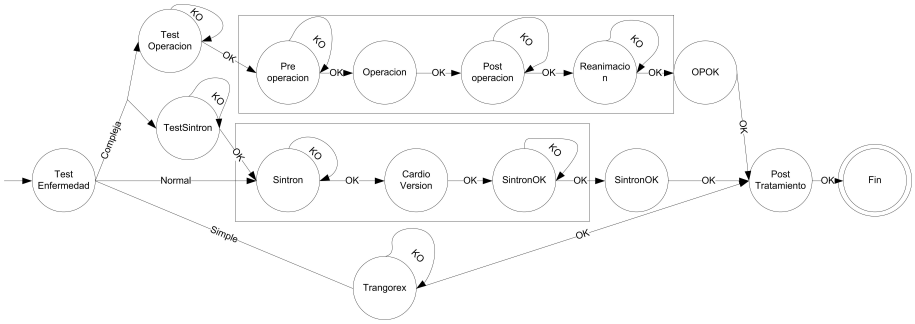

Figura 9.13: Estructura del Experimento IC_A

9.3.2. Experimento IC_B

El Experimento IC_B que se muestra en la Figura 9.13 es una modificación del Experimento IC_A que intenta probar como se comportan los algoritmos ante un patrón de ejecución complejo como pueda ser una rutina paralela entrelazada. En el experimento anterior se realizaban paralelamente una secuencia de operación y una secuencia de cardioversión que necesitaba la administración de Simtron. Sin embargo, realmente no se puede mantener en paralelo estas secuencias. esto es debido a que una persona que va a ser operada no puede tomar Simtron. En este caso hay que hacer una modificación en el experimento anterior para construir una rutina paralela entrelazada que ejecute la exclusión mutua entre ambas secuencias de modo que no se puedan ejecutar las dos a la vez.

El Experimento IC_B ha sido realizado utilizando un corpus creado a partir de la simulación del TPA definido mediante TPA Engine, como resultado se han obtenido 510 muestras que han sido utilizadas para inferir WF.

En la Tabla 9.9 se muestran los patrones seleccionados para su identificación en el Experimento IC_B. Es de notar que son muy parecidos a los del experimento anterior con la excepción de que añade tres patrones más. Los tres patrones añadidos con respecto al Experimento anterior son:

- *Bucle Test Simtron*: Este patrón se refiere al bucle de la actividad *TestSimtron* que es la actividad de entrada a la rutina de Sintron.

- *Bucle Test Operacion*: Este patrón se refiere al bucle de la actividad *TestOperacion* que es la actividad de entrada a la rutina de operación

- *Sincronización Compleja*: Este patrón se refiere al la transición paralela que une el final de la rutina *OPOK* y *SimtronOK*

En la Tabla 9.10 se puede observar los resultados de los coeficientes calculados de los distintos algoritmos.

El algoritmo PALIA en esta ocasión no ha conseguido identificar todos los patrones de WF. Sin embargo, ha obtenido una eficacia más alta que todos sus competidores (0,93).

	PALIA	HM			GPM			Alpha			Alpha++		
		I/F	I	F	I/F	I	F	I/F	I	F	I/F	I	F
Separación Inicial	X	-	-	-	X	-	-	X	-	-	-	-	-
Secuencia Simple	X	X	X	X	-	-	-	X	-	-	-	X	X
Bucle Trangorex	X	X	-	-	X	X	-	X	-	-	-	X	X
Secuencia Normal	X	-	-	-	X	-	-	X	-	-	-	X	X
Bucle Sintron	X	X	X	X	X	X	X	X	-	-	-	X	X
Bucle Eliminar Sintron	X	X	X	X	X	X	X	X	-	-	-	X	X
Secuencia Fin Sintron	X	X	X	X	X	X	X	X	-	-	-	-	-
Separación Compleja	X	-	-	-	-	-	-	X	-	-	-	-	-
Bucle Test Sintron	X	X	X	X	-	X	X	X	-	-	-	-	-
Bucle Test Operación	X	X	X	X	-	X	X	X	-	-	-	-	-
Secuencia Compleja	X	X	X	X	X	X	X	-	-	-	-	-	-
Bucle Posoperatorio	X	X	X	X	X	X	X	X	-	-	-	X	X
Bucle Reanimación	X	X	X	X	X	X	X	X	-	-	-	-	-
Sincronizacion Compleja	-	-	-	-	-	-	-	-	X	X	-	X	X
Secuencia Final	X	X	X	-	X	X	-	X	X	-	-	X	-

Tabla 9.9: Planilla de eficacia del Experimento IC_B

		Nodos	*Arcos*	Cf_{NxA}	*Patrones*	Cf_{Rm}	*Eficacia*	Cf_{LR}
Real		15	25	375	15	1,75	-	-
PALIA		15	35	525	14	3,2	0,93	1,50
Heuristic Miner	I/F	30	59	1770	11	1,97	0,73	6,44
	I	16	29	464	10	1,81	0,67	1,86
	F	14	27	378	9	1,93	0,60	1,68
Genetic Process Mining	I/F	30	54	1620	10	1,8	0,67	6,48
	I	16	29	464	10	1,86	0,67	1,86
	F	14	26	364	8	1,86	0,53	1,82
α	I/F	30	49	1470	13	3,33	0,87	4,52
	I	16	5	80	2	0,25	0,13	1,60
	F	14	3	42	1	0,14	0,07	1,68
α ++	I/F	-	-	-	-	-	-	-
	I	16	9	144	8	1,38	0,53	0,72
	F	14	7	98	7	1,43	0,47	0,56

Tabla 9.10: Resultados del Experimento IC_B

Solo un patrón no ha sido identificado por PALIA. Este patrón es el patrón de sincronización de las rutina paralela entrelazada. La legibilidad ha sido en este caso peor (1,50) que en los otros experimentos, aunque se mantiene en un orden bajo. En la Figura B.16 del anexo B se muestra el WF inferido por el algoritmo.

El algoritmo Heuristic Miner ha mejorado sensiblemente su eficacia (0,73) y mantenido la legibilidad (6,44) con respecto al Experimento IC_A. Esto es debido a que aunque haya una rutina paralela, al entrelazarse, su ejecución es secuencial, lo que facilita su identificación por los algoritmos basados en modelos de AFD. La Figura B.17 del anexo B muestra el WF inferido.

El algoritmo Genetic Process Mining también a tenido una buena mejora con respecto al Experimento IC_A, mejorando su eficacia (0,67) y su legibilidad (1,86). Al igual que el algoritmo HM, la naturaleza de los modelos AFD hace que funcionen mejor con rutinas paralelas entrelazadas. La Figura B.18 del anexo B muestra el WF inferido.

El Algoritmo α ha mantenido la eficacia (0,87) y ha mejorado la legibilidad (4,52). No obstante, el WF inferido es poco legible aun, porque se ha inferido utilizando corpus IF. El WF inferido se muestra en la Figura B.19 del anexo B.

El algoritmo α++ ha empeorado la eficacia(0,53) aunque ha mejorado la legibilidad sensiblemente (0,72). El WF inferido se muestra en la Figura B.20 del anexo B.

9.4. Conclusiones de los experimentos

De un modo general, PALIA ha resultado ser el tiene el mejor coeficiente de eficacia en todos los experimentos realizados. PALIA ha identificado casi todos los patrones y únicamente ha fallado en la identificación de dos patrones complejos uno referente a la sincronización de rutinas paralelas entrelazadas y el otro al sincronizar rutinas paralelas en otra rutina paralela. Por su parte la legibilidad de los WF inferidos por PALIA empeora cuando las secuencias paralelas se encuentran estructuradas y son largas aún así, la legibilidad es buena.

En cuanto al resto de los algoritmos, uno de los problemas más comunes en cuanto a la legibilidad que tienen los algoritmos basados en eventos es cuando se enfrentan a corpus IF. Cuando los algoritmos utilizan estos corpus duplican los arcos y nodos ya que tratan los eventos por separado, lo que hace que la legibilidad empeore mucho.

Por otra parte, los algoritmos basados en Redes de Petri α y $alpha++$ han demostrado debilidades con respecto a la identificación de separación y sincronización de secuencias paralelas. Funcionan mucho mejor ante secuencias simples que ante secuencias paralelas. Sin embargo, reaccionan positivamente al alargamiento de las secuencias paralelas. Sorprendentemente, el algoritmo α se ha mostrado más regular en los experimentos que su mejora $\alpha++$. En cuanto a la legibilidad, $\alpha++$ ha sido mejor que α siendo en algunos momentos mejor a la del modelo original, probablemente debido a que los patrones más complejos son tomados como ruido y sus estructuras son eliminadas.

Por último, los algoritmos no paralelos, HM y GPM, como era de esperar muestras una buena eficacia ante secuencias no paralelas, aunque tienen problemas para identificarlas entre secuencias paralelas. El algoritmo GPM se ha mostrado más sensible a el alargamiento de las secuencias paralelas que el HM. Por otro lado, cuando las secuencias paralelas se estructuras y se ejecutan de forma separada su eficacia aumenta mucho. El algoritmo GPM se ha mostrado más efectivo que el HM a lo largo de la experimentación

debido a que el HM suele eliminar muchos datos por considerarlos ruido. Por otro lado, en cuanto a la legibilidad, el algoritmo GPM se ha mostrado bastante regular a lo largo de la experimentación. Sin embargo, el algoritmo HM tiene una peor legibilidad que el GPM.

Tras la realización de los experimentos podemos decir que PALIA es el algoritmo que mejor coeficiente de eficacia ha conseguido en todos los experimentos. En cuanto a la legibilidad PALIA ha conseguido mantener la legibilidad a un buen nivel en todos los casos mientras que los demás algoritmos han sufrido problemas dependiendo de la naturaleza de los experimentos.

Parte III

Conclusiones y Principales Aportaciones

Capítulo 10

Conclusiones y trabajo futuro

A lo largo de esta memoria se han ido desgranando los problemas de Representación, Interpretación y Aprendizaje que surgen a la hora de implantar un sistema de WM para la estandarización de la Vías Clínicas. Para cada uno de estos problemas, se han presentado herramientas que permiten la aplicación de las tecnologías de WF en el campo de la gestión de Vías Clínicas basándose en la hipótesis y objetivos principales de la Tesis.

El propósito de este capítulo es revisar los objetivos y comprobar la consecución de estos. Para ello, se van a listar y analizar los resultados principales obtenidos a lo largo de la Tesis.

10.1. Conclusiones

El trabajo realizado a lo largo de la Tesis ha estado basado en la hipótesis y objetivos formulados al principio del documento. Específicamente, estas propuestas han ido abordando problemas de Representación, Interpretación y Aprendizaje de WF en general, aunque especialmente motivados para la resolución del problema de la estandarización de las Vías Clínicas. Según la metodología de presentación de la Tesis en este capítulo se aborda la evaluación del trabajo realizado y su correlación con los objetivos inicialmente marcados. De este modo, se va a proceder a la revisión de los resultados obtenidos en esta Tesis:

El objetivo principal de la Tesis fue definido en el preámbulo de la Tesis como sigue:

> Proponer un nuevo paradigma de Aprendizaje de Flujos de Trabajo basado en Actividades que utilice corpus que incorporen la información de inicio, fin y resultados de las acciones y aportar herramientas y algoritmos necesarios para implementar, evaluar y aprender sistemas basados en este paradigma.

Este objetivo ha sido cubierto mediante la presentación del paradigma de **Aprendizaje de Flujos de Trabajo orientados a Actividades**. Esta metodología implica la creación de corpus de aprendizaje que incorporan tanto la información de inicio y de fin de las acciones como los resultados de las propias acciones. Esta información permite a

los algoritmos de inferencia aprender no solo el flujo que siguen las acciones en un WF, sino también las razones por las que los cambios se realizan.

Para evaluar este paradigma se presentaron herramientas y algoritmos que resuelven los problemas de aplicación de este en cuanto a representación, interpretación y aprendizaje de WF.

En el campo de la Representación de WF se ha abordado la creación de un lenguaje expresivo y legible que sirva como modelo de representación de Vías Clínicas. Para ello, se enunció el siguiente objetivo secundario:

> Definir una herramienta de representación de Flujos de Trabajo acorde con el nuevo paradigma presentado, que permita representar fácilmente una gran cantidad de patrones de Flujos de Trabajo, que sea fácilmente legible, que sea fácilmente interpretable por sistemas de ejecución, y con una complejidad que facilite su inferencia utilizando técnicas de reconocimiento de patrones.

Para resolver este objetivo se ha desarrollado un **modelo de representación de WF, denominado TPA**, basado en el modelo de autómatas finitos con una buena legibilidad, capaz de representar correctamente las patrones de WF de flujo de datos de Van der Aalst, demostrando así su expresividad. Además, es capaz de representar el tiempo de una forma natural. Por otro lado, dada su complejidad gramatical regular permite utilizar un vasto marco teórico que facilita la utilización de técnicas de interpretación y aprendizaje para el apoyo al diseño de WF.

En cuanto a la interpretación de WF, se ha abordado la creación de un sistema de ejecución capaz de enriquecer la información de sus WLogs con los resultados de las acciones realizadas y que permitiera la modificación dinámica de las instancias de WF. Esto se especificó en el objetivo secundario:

> Implementar un motor de interpretación capaz de ejecutar el modelo de representación presentado tanto desde un punto de vista de guiado de procesos, como desde un punto de vista de simulación de procesos. Este motor de interpretación deberá ser capaz de generar muestras con los datos necesarios para resolver el paradigma de Aprendizaje de Flujos de Trabajo basado en Actividades.

Para desarrollar este objetivo, **se ha implementado un motor de ejecución de WF que permite ejecutar TPAs denominado TPAEngine**. Este motor es capaz de ejecutar los procesos cualquiera que fueran su patrones (paralelos o secuenciales) aprovechando el modelo del TPA. El TPAEngine no necesita grandes sistemas para funcionar, por lo que se puede ejecutar en maquinas con pocos recursos, manteniendo toda la expresividad de los TPA. Además debido a que el TPAEngine copia para cada instancia las reglas del proceso, estas pueden ser cambiadas en cualquier momento para cada instancia de una forma independiente

Para probar la validez del ABWM se ha estudiado la implementación de un algoritmo capaz de utilizar la información de esta metodología para inferir WF que sirvan para apoyar al diseño de Vías Clínicas. En esta linea se en enunció objetivo secundario:

> Proponer un nuevo algoritmo de inferencia en el marco del paradigma de Aprendizaje de Flujos de Trabajo basado en Actividades.

Para abordar este objetivo se ha diseñado e implementado un **algoritmo de inferencia de WF llamado PALIA** acorde con la metodología del ABWM. Este algoritmo es capaz de inferir WF expresados en forma de TPA a partir de muestras producidas por motores de WF como el TPAEngine, que incorporan en sus WLogs la información de inicio y de fin de las acciones y los datos referidos a los resultados de estas. Sin embargo, este algoritmo no se limita únicamente a funcionar como algoritmo orientado a actividades, sino que es capaz de inferir los flujos de WF como si fuera un algoritmo orientado a eventos.

Para poder evaluar la capacidad de inferencia de este algoritmo ha sido necesario abordar la creación de un marco de evaluación de sistemas de inferencia de WF aplicados al diseño de Vías Clínicas, lo que desembocó en el objetivo secundario:

> Proponer nuevas métricas para la validación de algoritmos de Aprendizaje de Flujos de Trabajo que permitan la comparación entre modelos basados en actividades y modelos basados en eventos.

Para resolver este objetivo se han desarrollado **una nueva métrica de evaluación de la capacidad de inferencia de los algoritmos de WM** basadas en el calculo la eficacia en la capacidad de inferencia expresiva y el calculo de la legibilidad relativa al numero de patrones aceptados. Esta métrica ha sido probada con PALIA en comparación a los algoritmos de WM existentes en la literatura utilizando tanto corpus existentes del EBWM, como corpus basados en Vías Clínicas. En estos experimentos PALIA ha obtenido los mejores resultados en eficacia manteniendo siempre la legibilidad en un buen nivel.

La metodología ABWM junto con las herramientas de Representación, Interpretación y Aprendizaje, se han desarrollado basándose en la hipótesis:

> La utilización de métodos de inferencia inductiva aplicados a la estandarización de los procesos involucrados en una Vía Clínica a partir de corpus basados en los datos de inicio, fin y resultados de las acciones realizadas en los protocolos de cuidado de los pacientes permite el aprendizaje de modelos que describan el flujo de las actividades y las reglas de cambio que explican su funcionamiento

Los experimentos realizados en esta Tesis muestran, de manera acorde con la hipótesis, que **es posible inferir WF que describan el flujo de los procesos en ejecución junto con las reglas de cambio de estos utilizando técnicas de inferencia inductiva** basadas en inferencia gramatical mediante el algoritmo de aprendizaje (**PALIA**), que aprende modelos (basados en **TPA**) a partir de muestras que incorporan datos de las acciones referentes al inicio, fin y resultado de las acciones (de manera acorde al **ABWM**) a partir de la interpretación de Vias Clínicas (utilizando el **TPAEngine**)

10.2. Trabajo Futuro

En esta Tesis se ha abordado la aplicación de técnicas de WM para el apoyo a la estandarización de Vías Clínicas. Sin embargo, las herramientas descritas a lo largo del documento pueden ser aplicadas a problemas similares en el campo de la estandarización de procesos. Además, existen puntos sobre la representación e interpretación de procesos que no han sido objetivo de esta Tesis pero que pueden aprovecharse de los resultados alcanzados en ella. Entre las líneas de futuro que se esperan cabe destacar:

- La utilización de sistemas gráficos para representar Vías Clínicas basadas en WF permite a los médicos visualizar y modificar los procesos de cuidado de una forma sencilla. En esta Tesis se ha abordado la representación formal y se han dado bases para la representación gráfica, pero no se ha creado un sistema gráfico de diseño de Vías Clínicas. En esta linea, actualmente se está creando una aplicación software capaz de expresar WF de una forma visual, basada en el modelo del TPA, que permita a los expertos en WF diseñar y evaluar los resultados de los algoritmos de inferencia utilizando toda la potencia de las utilidades gráficas.

- Para poder aplicar el ABWM, es necesario implantar sistemas de interpretación de WF y de este modo recoger los datos necesarios que permiten la utilización de este tipo de algoritmos. En este punto los resultado obtenidos en esta Tesis ya se están aplicando en proyectos europeos que persiguen el modelado del comportamiento humano y los sistemas de inteligencia ambiental, como el proyecto VAALID [4] o proyectos que buscan la representación de modelos de planes de cuidados para pacientes crónicos como el Proyecto Heart Cycle [3]. En esta línea, en un futuro se pretende llevar a producción estos sistemas y aplicar estas técnicas a plataformas reales de cuidado como una herramienta más de Data Mining para el apoyo a la gestión de los cuidados. En este aspecto se han realizado contactos con la empresa ITDeusto, y el Hospital La Fe de Valencia.

- El modelado de comportamiento humano es un complejo problema que actualmente está en investigación. Este campo pretende descubrir patrones de comportamiento en las personas que ayuden a comprender mejor su conducta. Dada la gran variabilidad y continua evolución del comportamiento humano la utilización de modelos deductivos es muy complicada en este campo. Utilizando técnicas de ABWM podremos inferir modelos que puedan ser utilizados para definir comportamientos comunes de personas de conductas similares. Esto nos permitirá crear clasificadores que pueden ser utilizados por ejemplo para detectar problemas como el Alzheimer, la Demencia Senil, el Parkinson, depresiones u otras enfermedades en etapas muy iniciales. El

proyecto PERSONA [1] pretende crear una plataforma de Inteligencia Ambiental que actúe automáticamente ante determinadas situaciones en pos de ayudar a los usuarios del sistema. Por ello, en PERSONA se está estudiando la utilización de clasificadores basados en ABWM para detectar conductas que requieran modificación. En este ámbito se han publicado artículos donde se aplica el ABWM al modelado del comportamiento humano [18, 19].

- Actualmente, es común que existan sistemas viajen con los propios usuarios ejecutándose en los propios dispositivos móviles que los acompañan. Esto permite a los sistemas tener un control holístico de los procesos facilitando que los estos se encuentren disponibles en todo momento. La creación de motores de interpretación de WF para dispositivos móviles que aprovechen la eficiencia del modelo del TPA es un punto de evolución importante en la interpretación de WF. Esto permitirá que los WF puedan viajar con los actores de los procesos, aumentando la sue eficacia y permitiendo la estandarización de los procesos de un modo más global.

- En esta Tesis se ha abordado la ejecución de WF desde el punto de vista de la orquestación de procesos. Este modelo asume sistema centralizado basado en un motor que dirige las acciones que se ejecutan. Sin embargo existe otra aproximación basada en la ejecución distribuida de los procesos llamada coreografía de procesos. Los modelos de coreografía se basan en la autonomía de las acciones en sí misma donde la función del coreógrafo es coordinar las reglas de comunicación entre los servicios. Estas reglas son susceptibles de ser inferidas para descubrir modelos menos acoplados y distribuidos y que permitan una mayor independencia que los modelos orquestados. En esta línea se está realizando una Tesis doctoral que estudia este tipo de sistemas aplicando el TPA para definir los servicios autónomos.

Capítulo 11

Principales aportaciones originales de la Tesis

En esta Tesis se han aportado soluciones a problemas que no cubren los modelos actuales de WM. De este modo se han creado y publicado metodologías (ABWM), esquemas de representación (TPA), motores de interpretación (TPAEngine) y algoritmos de inferencia(PALIA) para apoyar la resolución de problemas tales como el diseño de Vías Clínicas. Seguidamente se procederá a enumerar las aportaciones originales realizadas en cada uno de los puntos tratados en este trabajo en forma de publicaciones científicas asociadas al trabajo realizado de la Tesis, proyectos de investigación donde se ha aplicado los conceptos estudiados y aplicaciones software que han sido implementadas durante el desarrollo de la Tesis.

11.1. Publicaciones Asociadas

En el marco de esta Tesis se han publicado los siguientes trabajos:

Publicaciones en Capítulos de Libros

Una extensión del TPA como herramienta para la para la definición de WF que definan protocolos del guiado del estilo de vida, llamada LAP, se publicó en:

> [44] David Dominguez, **Carlos Fernandez**, Teresa Meneu, Juan Bautista Mocholi, and Riccardo Seraffin. Medical guidelines for the patient: Introducing the life assistance protocols. In *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*, volume 139, page 282. IOS Press, 2008

Publicaciones en Revistas

Un trabajo donde se hablaba en la creación de protocolos médicos extraídos de Vías Clínicas integrados dentro de aplicaciones de telemedicina se publicó en la prestigiosa revista:

[60] Sergio Guillen, Teresa Arredondo, Vicente Traver, Juan Miguel García, and **Carlos Fernandez**. Multimedia telehomecare system using standard tvset. *IEEE Transactions on Biomédical Engineering ISSN:0276-6547 22 Citas JCR:1.665*, 49:1431–1437, 2002

En cuanto a la aplicación de modelos de automatización de las Vías Clínicas, se han realizado varios artículos a revistas donde se estudia la aplicación modelos de WF y el guiado de los procesos de cuidado mediante Vías Clínicas automatizadas:

[101] Vicente Traver, **Carlos Fernandez**, Juan Carlos Naranjo, Eduardo Monton, Sergio Guillen, and Bernardo Valdivieso. Sistemas m-health: la solución para las necesidades de una unidad de hospitalización a domicilio. *I+S Informatica y Salud (ISSN 1579-8070)*, 44:43–49, 2004

[102] Vicente Traver, Susana Pomés, **Carlos Fernandez**, José Conca, Lucas Sanjuán, Eduardo Roldan, Javier Berrio, Maria José Nodal, Beatriz Albella, Manuel Perez, and Sergio Guillen. Lyra: Sistema de gestión integrado para una unidad de hospitalización a domicilio. *I+S Informatica y Salud: Especial Comunicaciones en Sanidad (ISSN 1579-8070)*, 61:20–24, 2007

[73] Juan Carlos Naranjo, **Carlos Fernandez**, Susana Pomes, and Bernardo Valdivieso. Care-paths: Searching the way to implement pathways. *Computers in Cardiology ISSN:0276-6547*, 33:285–288, 2006

La utilización de PALIA como algoritmo de ABWM para la inferencia de Vías Clínicas fue presentada en:

[16] **Carlos Fernandez** and Jose Miguel Benedí. Timed parallel automaton learning in workflow mining problems. In *Ciencia y Tecnología en la Frontera ISSN:1665-9775*, 2008

En cuanto a la interpretación de WF, el TPAEngine fue presentado como herramienta de interpretación de WF en:

[21] **Carlos Fernandez**, Carlos Sanchez, Vicente Traver, and Jose Miguel Benedí. Tpaengine: Un motor de workflows basado en tpas. In *Ciencia y Tecnología en la Frontera ISSN:1665-9775*, 2008

Una solución para el problema del modelado del comportamiento humano mediante una metodología basada en ABWM fue presentada en:

[19] **Carlos Fernandez**, Juan Pablo Lazaro, Gema Ibañez, and David Dominguez. Workflow mining para el modelado individualizado del comportamiento humano en el contexto de la inteligencia ambiental. In *Ciencia y Tecnología en la Frontera ISSN:1665-9775*, 2008

Publicaciones en Congresos

La metodología de ABWM y el TPA como herramienta de representación de WF fue presentada en:

[17] **Carlos Fernandez** and Jose Miguel Benedi. Activity-based workflow mining: A theoretical framework. In *Workshop On Technologies for Healthcare and Healthy Lifestyle ISBN:978-84-611-1311-8*, 2006

Diversos artículos en congresos sobre la aplicación de Vías Clínicas a unidades de Hospitalización a domicilio fueron presentados en:

[75] Juan Carlos Naranjo, **Carlos Fernandez**, Salvador Vera, Sergio Guillen, and Bernardo Valdivieso. Proyecto ideas: Systema de informacion distribuido para una unidad de hospitalización domiciliaria. In *Congreso Nacional de Informatica para la Salud(INFORSALUD)*, 2002

[20] **Carlos Fernandez**, Juan Carlos Naranjo, Eduardo Monton, Vicente Traver, and Bernardo Valdivieso. Gestion de hospitalizacion a domiclio distribuida mediante servicios web. In *Congreso Anual de la Sociedad Española de Ingeniería Biomedica (CASEIB)*, 2003

Una implementación de un motor de interpretación de LAPS basada en sistemas multiagente fue presentada en:

[71] Juan Bautista Mocholi, David Dominguez, and **Carlos Fernandez**. Towards an environment under which executing laps. In *Workshop On Technologies for Healthcare and Healthy Lifestyle ISBN:978-84-611-1311-8*, 2006

Una arquitectura de validación de modelos AAL con modelado del usuario virtual mediante ABWM ha sido presentada en:

[74] Juan Carlos Naranjo, **Carlos Fernandez**, Maria Pilar Sala, Juan Bautista Mocholi, Michael Hellenschmidt, and Franco Mercalli. A modelling framework for ambient assisted living validation. In *Human Computer Interaction International 2009. Aceptado, Pendiente de publicación*, 2009

Se han presentado resultados de pruebas del algoritmo PALIA para el modelado del Comportamiento Humano mediante inferencia en:

[18] **Carlos Fernandez**, Juan Pablo Lazaro, and Jose Miguel Benedi. Workflow mining application to ambient intelligence behaviour modelling. In *Human Computer Interaction International 2009. Aceptado, Pendiente de publicación*, 2009

11.2. Proyectos Asociados

El trabajo realizado a lo largo de esta Tesis ha probado en diferentes proyectos de investigación co-financiados por distintas entidades. Entre estos proyectos se encuentran proyectos nacionales financiados por el Ministerio de Ciencia y Tecnología y proyectos de la Union Europea financiados dentro del V, VI y VII programa marco donde se han contado y se cuentan entre otros con partners Estratégicos como la multinacional Philips, la Consellería de Sanitat de la Generalitat Valenciana o el instituto Fraunhofer de Alemania, y centros hospitalarios como el Hospital la Fe de Valencia, la Ospederia de Parma, el Hospital San Raffaelle de Milan o el Hospital Virgen de los Lirios de Alcoy.

Entre estos proyectos destacan:

- El proyecto **IDEAS in e-Health** [5] (Integrated Distributed Environment for Application Services in e-Health. V Framework Program IST Project 34614) abordó el problema de la creación de una plataforma integral de gestión médica mediante el Modelo ASP (Application Services Provider).

  En el ámbito de la Tesis, dentro de este proyecto se ofrecieron soluciones al problema de la definición de protocolos de cuidado creando una herramienta para la automatización y guiado de Vias Clínicas sobre una unidad de hospitalización a domicilio en colaboración con el Hospital La Fe de Valencia.

- **AGORA2000** [2] (Innovative IST Platforms and Services to Support a Democratic Regional/Urban Planning Process. V Framework Program IST Project 20982) fue un proyecto dedicado a la creación de plataformas y servicios para mejorar los sistemas de gestión electrónica de los planes de desarrollo urbano de los ayuntamientos y regiones.

  En el ámbito de la Tesis, se realizó un trabajo para automatizar el WF de los planes de actuación urbanística del Ayuntamiento de Valencia, para ayudar a la automatización de los procesos de presentación, aprobación y exposición pública de dichos planes.

- El proyecto **CAREPATHS** [6] (CAREPATHS An intelligent support environment to improve the quality of decision processes in health communities. VI Framework Program IST Project 507017) estuvo dedicado a la creación e implantación de Vías Clínicas en los procesos de decisión hospitalaria.

  En el ámbito de la tesis, se creó un sistema de WM que recoge datos de ejecuciones pasadas basadas en actividades para guiar a los médicos en los procesos de cuidado de los pacientes y en el diseño de WF. Este trabajo fue presentado en el capitulo 6 del deliverable D3.2 [26] de este proyecto. Además, se contribuyó al diseño de una ontología para definir Vías Clínicas publicada en el capítulo 4 de este deliverable.

- En el proyecto **MyHeart** [8] (MyHeart, Fighting cardio-vascular diseases by prevention and early diagnosis. VI Framework Program IST Project 507816)

  En el ámbito de la Tesis, se realizó el diseñó e implementación de un sistema de orquestación de procesos basados en autómatas finitos para la coordinación de procesos, llamado PIWE (del inglés *Professional Interaction Workflow Engine*) realizados

por *third parties*. Este sistema se presentó en el deliverable D2 [28] de este proyecto, y su implementación en el deliverable D7 [29] de este proyecto.

- El proyecto **LYRA** [42] financiado por el Ministerio de Ciencia y Tecnología dentro del proyecto Alcoy Ciudad Digital se realizó la implantación de una plataforma de gestión de cuidados distribuidos para una unidad de Hospitalización a Domicilio.

  En el ámbito de la Tesis, se implantó el modelo diseñado en el proyecto IDEAS in e-Health en el Hospital Virgen de los Lírios de Alcoy como proyecto piloto para su extensión al resto de los hospitales de la Comunidad Valenciana.

- El proyecto **PIPS** [7] (Personalised Information Platform for life and health Services. VI Framework Program IST Project 507019) esta orientado a la creación de una plataforma personalizada de gestión de servicios para la prevención en el ámbito de la salud aplicada sobre ciudadanos sanos.

  En el ámbito de la Tesis, se trabajó en la especialización del TPA para la creación de un modelo de representación de protocolos para el flujo de la vida diarias de las personas llamados *Life Activity Protocols*(LAP) y se definió un motor de ejecución de LAPs utilizando sistemas multiagente.

- El proyecto **PERSONA** [1] (PERceptive Spaces prOmoting iNdependent Aging. VII Framework Program IST Project 045459) tiene como objetivo la creación de una plataforma de Inteligencia Ambiental orientada al paradigma del Ambient Assisted Living (AAL) de tal manera que proporcione una solución que promueva el envejecimiento independiente.

  En el ámbito de la Tesis, se está creando un modelo de orquestación de servicios basada en la utilización de sistemas gráficos basados en la definición del TPA ejecutandose sobre jBPM. Este módulo está pensado para la ejecución de servicios en un entorno de Inteligencia Ambiental. Este trabajo está publicado en el deliverable D3.3.1 [30] de este proyecto.

- El proyecto **HeartCycle** [3] (Heart Cycle Project: Compliance and effectiveness in HF and CHD closed-loop management. VII Framework Program IST Project 216695) tiene como objetivo la creación de una plataforma para la gestión del cuidado de las enfermedades crónicas del corazón desde un punto de vista holístico, aportando herramientas que permitan a los médicos tener un control total sobre las acciones y recomendaciones que el paciente sigue durante el desarrollo de su enfermedad.

  En el ámbito de la Tesis, se va a utilizar el TPA como herramienta para la definición y ejecución de Vías Clínicas para la gestión de enfermos de condiciones crónicas ejecutado sobre un motor de interpretación basado en jBPM.

- El proyecto **VAALID** [4] (VAALID Project: Accessibility and usability validation framework for AAL interaction design process. VII Framework Program IST Project 224309) está pensado para definir un marco de validación de entornos AAL mediante sistemas 3D para permitir a los diseñadores de sistemas probar sus entornos antes de llevarlos a producción.

  En el ámbito de la Tesis, se está trabajando en la definición del comportamiento de los servicios, sensores y usuarios virtuales del sistema de validación de sistemas AAL

mediante TPA. Además se utilizará PALIA para la inferencia del comportamiento del usuario virtual mediante técnicas de ABWM.

11.3. Software Desarrollado

Un importante resultado de esta Tesis es el Software generado durante su desarrollo. Este software complementa la investigación presentada y pone en practica los conceptos teóricos especificados. De entre el software desarrollado en esta Tesis cabe destacar:

- El **TPAEngine** es un motor de interpretación de WF que permite no solo la ejecución de WF basados en TPA, sino que permite su simulación para la prueba del flujo de ejecución de los procesos y la creación de corpus de WM.
- El algoritmo de inferencia **PALIA** ha sido desarrollado formando parte de una herramienta completa que utiliza como entrada WLogs para la inferencia de WF en formato TPA y reproduce los resultados de la inferencia de una forma gráfica.
- Aparte, se han desarrollado diversas utilidades secundarias como:
  - Conversores de TPA a lenguaje DOT, que permiten la representación gráfica estándar de los resultados y su utilización por parte de herramientas gráficas compatibles con el lenguaje DOT
  - Módulos de calculo de los coeficientes de legibilidad a partir de ficheros dot, lo que permite no solo calcular la legibilidad de los resultados de PALIA, sino también la legibilidad de los resultados de los algoritmos implementados en ProM.
  - Conversores de WLogs del TPAEngine a formato ProM, que permite a los algoritmos implementados en esta herramienta utilizar los WLogs que creados por el TPAEngine.

Parte IV

Anexos

Apéndice A

Patrones de Control de Flujo y su Representación formal mediante TPAs

En este Anexo se pretende demostrar como el TPA es capaz de cumplir la expresividad requerida en los 20 patrones de Van der Aalst. Para ello se van a enumerar y describir cada uno de estos patrones, probándose la expresividad del autómata mediante ejemplos, representados tanto de una forma gráfica, como formal.

Los veinte patrones definidos se presentan en seis grupos diferentes que son los siguientes:

A.1. Patrones de flujo básico

Los patrones de flujo básico son aquellos que tratan aspectos básicos de los procesos de control. Estos patrones aparecen listados a continuación:

Patrón 1: Secuencia

El patrón de Secuencia describe el proceso más simple dentro de un flujo de control. Este patrón expresa el proceso en el que una actividad se inicia después de haberse completado de otra en el mismo proceso.

En la Figura A.1 se puede un ejemplo de como se define este patrón gráficamente con el TPA. Una representación formal del ejemplo se presenta a continuación:

$$\text{A} = \{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$$
$$\text{C} = \{c^{\infty}\},$$

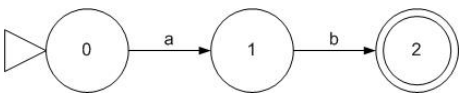

Figura A.1: Patrón de Secuencia

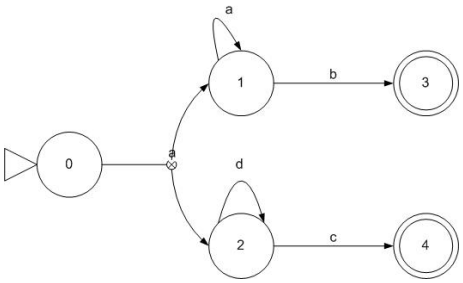

Figura A.2: Patrón de Separación paralela

P = {0,1,2},
N = $\{n_0 : [0, c^\infty], n_1 : [1, c^\infty], n_2 : [2, c^\infty]\}$,
Q = $\{q_0 : n_0, q_1 : n_1, q_2 : n_2\}$,
T = $\emptyset$,
$\Phi = \{a, b\}$,
$\Sigma = \{a, b, \lambda\}$,
$\gamma : \{[n_0, a, n_1], [n_1, b, n_2]\}$,
$\delta : \{[q_0, a, q_1], [q_1, b, q_2]\}$,
$q_0 = \{q_0\}$,
$F = \{q_2\}$ □

Como vemos, esta representación formal es muy simple. Las funciones γ y δ se refieren con los mismos arcos a los mismos nodos. Estos arcos definen la secuencia definida en el autómata.

Patrón 2: Separación Paralela

El patrón de separación paralela describe un proceso en el que desde una actividad completada, se da lugar a la ejecución de dos o más actividades en paralelo.

En la Figura A.2 se puede observar un ejemplo del patrón. En esta figura, desde el estado inicial 0 se pasa, utilizando el símbolo a, a los estados 1 y 2 concurrentemente. Una representación formal del ejemplo se puede observar a continuación:

A =$\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C = $\{c^\infty\}$,
P = {0,1,2,3,4},
N = $\{n_0 : [0, c^\infty]$, $n_1 : [1, c^\infty]$, $n_2 : [2, c^\infty]$, $n_3 : [3, c^\infty]$, $n_4 : [4, c^\infty]\}$,
Q = $\{q_0 : n_0$, $q_{12} : n_1 n_2$, $q_{23} : n_2 n_3$, $q_{14} : n_1 n_4$, $q_{34} : n_3 n_4\}$,
T = $\emptyset$,
$\Phi = \{a, b, c, d\}$,
$\Sigma = \{a, b, c, d, \lambda\}$,

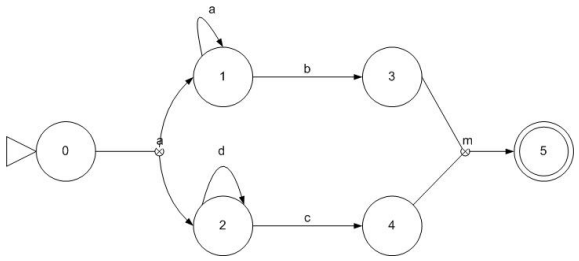

Figura A.3: Patrón de Sincronización

γ: $\{[n_0, a, n_1 n_2]$, $[n_1, a, n_1]$, $[n_2, d, n_2]$, $[n_1, b, n_3]$, $[n_2, c, n_4]$ $\}$,
δ: $\{[q_0, a, q_{12}]$, $[q_{12}, a, q_{12}]$, $[q_{12}, d, q_{12}]$, $[q_{12}, b, q_{23}]$, $[q_{12}, c, q_{14}]$, $[q_{23}, c, q_{34}]$, $[q_{14}, b, q_{34}]\}$,
$q_0 = \{q_0\}$,
$F = \{q_{34}\}$ □

Esta representación formal pone de manifiesto las diferencias entre Nodos y Estados, mientras los nodos son actividades los estados son un conjunto de ellas. La función γ muestra como se producen las transiciones a nivel de nodo, tal y como se muestra en la Figura A.2, sin embargo, la función δ muestra las transiciones entre los estados, contando todas las posibilidades de transición entre todos los nodos que se pueden ejecutar concurrentemente.

Patrón 3: Sincronización

El patrón se sincronización viene asociado a procesos que tienen en ejecución multiples actividades ejecutadas en paralelo. En algunos casos, es necesario que un subconjunto de las actividades que se están ejecutando en paralelo terminen antes seguir ejecutando el WF. El elemento sincronizador, esperará a que todas las actividades terminen y continuará el proceso con la siguiente o las siguientes actividades.

En la Figura A.3 se puede ver un ejemplo de representación del patrón de sincronización. En el gráfico se puede ver un proceso de separación paralela que divide el proceso inicial. El patrón de sincronización juntará las dos secuencias de actividades en una. Una representación formal de este patrón se define a continuación:

A =$\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C = $\{c^\infty\}$,
P = {0,1,2,3,4,5},
N = $\{n_0 : [0, c^\infty]$, $n_1 : [1, c^\infty]$, $n_2 : [2, c^\infty]$, $n_3 : [3, c^\infty]$, $n_4 : [4, c^\infty]$, $n_5 : [5, c^\infty]\}$,
Q = $\{q_0 : n_0$, $q_{12} : n_1 n_2$, $q_{23} : n_2 n_3$, $q_{14} : n_1 n_4$, $q_{34} : n_3 n_4$, $q_5 : n_5\}$,
T = $\emptyset$,
$\Phi = \{a, b, c, d, m\}$,

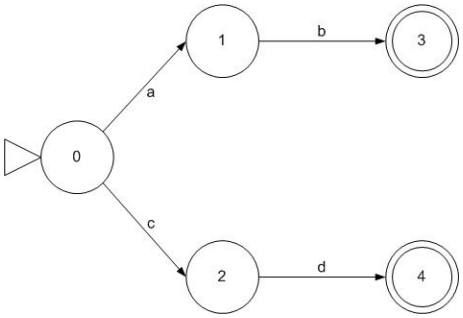

Figura A.4: Patrón de Elección exclusiva

$\Sigma = \{a, b, c, d, m, \lambda\}$,
γ: $\{[n_0, a, n_1 n_2]$, $[n_1, a, n_1]$, $[n_2, d, n_2]$, $[n_1, b, n_3]$, $[n_2, c, n_4]$, $[n_3 n_4, m, n_5]$ $\}$,
δ: $\{[q_0, a, q_{12}]$, $[q_{12}, a, q_{12}]$, $[q_{12}, d, q_{12}]$, $[q_{12}, b, q_{23}]$, $[q_{12}, c, q_{14}]$, $[q_{23}, c, q_{34}]$, $[q_{14}, b, q_{34}]$, $[q_{34}, m, q_5]\}$,
$q_0 = \{q_0\}$,
$F = \{q_5\}$ □

En esta representación formal se puede ver como se definen los patrones de separación y sincronización. Tanto en la función γ como en la función δ se pasa desde los estados 3 y 4 donde terminan las secuencias de actividades concurrentes, con el símbolo m al estado final 5, solamente cuando los estados 3 y cuatro hayan sido alcanzados. Si se hubiese deseado que el paso hubiese sido inmediato cuando hubiesen acabado las dos, habría bastado con utilizar relojes c^0 en los estados finales de las secuencias de actividades y transiciones t^0 en los arcos.

Patrón 4: Elección exclusiva

El patrón de Elección exclusiva permite a los WFs realizar decisiones a la hora de ejecutar actividades en función de las ramas escogidas.

En la Figura A.4 se puede ver como se define el patrón de elección exclusiva con un TPA en un ejemplo. El paso desde el estado 0 al estado 1 se produce cuando llegue un símbolo a, mientras que el paso del estado 0 al 2 se produce cuando llega un símbolo c. Estos símbolos pueden generados automáticamente por un motor al finalizar la actividad del estado 0 o pueden ejecutarse por un evento externo. En este caso, este patrón define un paso automático, el patrón que define el mismo caso de manera externa será abordado más adelante. La representación formal del ejemplo se presenta a continuación:

A $=\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C $= \{c^\infty\}$,
P $= \{0,1,2,3,4,5\}$,

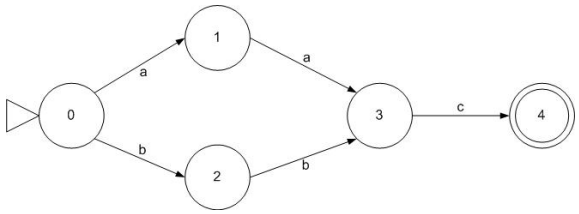

Figura A.5: Patrón de Fusión Simple

$N = \{n_0 : [0, c^\infty], n_1 : [1, c^\infty], n_2 : [2, c^\infty], n_3 : [3, c^\infty], n_4 : [4, c^\infty]\},$
$Q = \{q_0 : n_0, q_1 : n_1, q_2 : n_2, q_3 : n_3, q_4 : n_4\},$
$T = \emptyset,$
$\Phi = \{a, b, c, d\},$
$\Sigma = \{a, b, c, d, \lambda\},$
$\gamma : \{[n_0, a, n_1], [n_1, b, n_3], [n_0, c, n_2], [n_2, d, n_4]\},$
$\delta : \{[q_0, a, q_1], [q_1, b, q_3], [q_0, c, q_2], [q_2, d, q_4]\},$
$q_0 = \{q_0\},$
$F = \{q_3, q_4\}$ □

Como podemos ver, es muy simple representar este patrón formalmente utilizando TPAs, simplemente, agregando un arco por cada elección que decidamos.

Patrón 5: Fusión simple

El patrón de fusión simple viene asociado a procesos donde hay caminos alternativos de ejecución que confluyen en un estado común. La separación de los caminos se puede definir usando el patrón de elección exclusiva, mientras que la fusión de los caminos en la misma rama se realiza utilizando el patrón de fusión simple. Este patrón supone que ninguna de las ramas que junta puede ser ejecutada en paralelo, estos casos serán abordados en otros patrones.

En la Figura A.5 se define un ejemplo patrón de fusión simple. En este caso se produce una elección exclusiva de dos ramas que confluyen de los estados 1 y 2 al estado 3. Una representación formal se presenta a continuación:

$A = \{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
$C = \{c^\infty\},$
$P = \{0,1,2,3,4\},$
$N = \{n_0 : [0, c^\infty],\ n_1 : [1, c^\infty],\ n_2 : [2, c^\infty],\ n_3 : [3, c^\infty],\ n_4 : [4, c^\infty]\},$
$Q = \{q_0 : n_0,\ q_1 : n_1,\ q_2 : n_2,\ q_3 : n_3,\ q_4 : n_4\},$
$T = \emptyset,$
$\Phi = \{a, b, c\},$
$\Sigma = \{a, b, c, \lambda\},$
γ: $\{[n_0, a, n_1],\ [n_0, b, n_2],\ [n_1, a, n_3],\ [n_2, b, n_3],\ [n_3, c, n_4]\ \},$

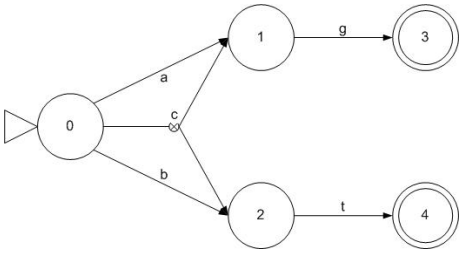

Figura A.6: Patrón de Multielección

δ: $\{[q_0, a, q_1], [q_0, b, q_2], [q_1, a, q_3], [q_2, b, q_3], [q_3, c, q_4]\}$,
$q_0 = \{q_0\}$,
$F = \{q_3, q_4\}$ □

Como podemos ver es muy simple la creación de éste patrón simplemente añadiendo arcos al estado común. Utilizando λ-transiciones o relojes c^0, sería posible realizar pasos entre actividades instantaneamente, en lugar de utilizar símbolos.

A.2. Patrones de Ramificación Avanzada y Sincronización

Aunque el conjunto de patrones de ramificación avanzada y sincronización no pertenecen al conjunto de patrones básicos por su complejidad, esto no significa que su uso sea minoritario. Este conjunto de cuatro patrones es bastante común en la definición de procesos de la vida real. En esta sección se procederá a su definición:

Patrón 6: Multielección

El patrón de multielección propone una mejora del patrón de elección exclusiva. Mientras que el patrón de elección exclusiva sólo permitía ejecutar una secuencia de actividades a la vez, el patrón de multielección permite la ejecución de un número indeterminado de secuencias dependiendo de la selección en tiempo de ejecución.

En la Figura A.6 se presenta un ejemplo de patrón de multielección. El paso del estado 0 a los estados 1 y 2 esta regulado por tres arcos, uno, marcado con el símbolo a que lleva al estado 1, otro, marcado con el símbolo b que lleva al estado 2, y un tercero, marcado con el símbolo c que lleva a la ejecución en paralelo de ambos estados. Cada arco marca una posibilidad de ejecución del proceso, marcado por el símbolo que utilicemos. Una representación formal del ejemplo es presentada a continuación:

A $=\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C $= \{c^\infty\}$,

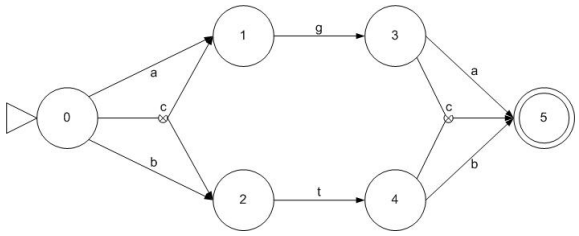

Figura A.7: Patrón de Fusión Sincronizada

P = {0,1,2,3,4},
N = $\{n_0 : [0, c^\infty]$, $n_1 : [1, c^\infty]$, $n_2 : [2, c^\infty]$, $n_3 : [3, c^\infty]$, $n_4 : [4, c^\infty]\}$,
Q = $\{q_0 : n_0$, $q_1 : n_1$, $q_2 : n_2$, $q_3 : n_3$, $q_4 : n_4$, $q_{12} : n_1 n_2$, $q_{34} : n_3 n_4$, $q_{14} : n_1 n_4$, $q_{23} : n_2 n_3\}$,
T = $\emptyset$,
$\Phi = \{a, b, c, g, t\}$,
$\Sigma = \{a, b, c, g, t, \lambda\}$,
γ: $\{[n_0, a, n_1]$, $[n_0, b, n_2]$, $[n_0, c, n_1 n_2]$, $[n_1, g, n_3]$, $[n_2, t, n_4]\}$,
$\delta : \{[q_0, a, q_1]$, $[q_0, b, q_2]$, $[q_0, c, q_{12}]$, $[q_1, g, q_3]$, $[q_2, t, q_4]$, $[q_{12}, t, q_{14}]$, $[q_{12}, g, q_{23}]$, $[q_{14}, g, q_{34}]$, $[q_{23}, t, q_{34}]$ },
$q_0 = \{q_0\}$,
$F = \{q_{34}\}$ □

La representación formal muestra como se define un grupo de estados diferente para cada posible ejecución del WF. El automata final utilizará por tanto, estados diferentes para ejecuciones diferentes.

Patrón 7: Fusión sincronizada

El patrón de función sincronizada funciona de manera análoga al patrón de multielección, sólo que al final de un conjunto de secuencias de actividades. Cuando en un proceso se ejecuta un patrón de multielección, es más que probable que en algún momento el conjunto de las secuencias de actividades potencialmente seleccionables para ser ejecutadas confluyan en un mismo punto. Es este caso el principal problema es saber como han de continuar, es decir, que estados hay que fusionar. El patrón de fusión sincronizada define el proceso de fusión de varios hilos de secuencias de actividades en uno, sincronizandolas según se hayan ejecutado.

En la Figura A.7 se muestra un ejemplo del patrón de fusión sincronizada. En el ejemplo podemos ver como el proceso separa la secuencia inicial en dos secuencias de actividades conforme a los símbolos recibidos y, posteriormente, los vuelve a juntar correspondientemente a como hayan sido ejecutados. Una representación formal del problema se muestra a continuación:

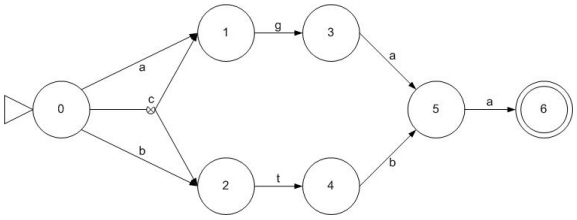

Figura A.8: Patrón de Fusión Multiple

A $=\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C $= \{c^{\infty}\}$,
P = {0,1,2,3,4,5},
N $= \{n_0 : [0, c^{\infty}],\ n_1 : [1, c^{\infty}],\ n_2 : [2, c^{\infty}],\ n_3 : [3, c^{\infty}],\ n_4 : [4, c^{\infty}],\ n_5 : [5, c^{\infty}]\}$,
Q $= \{q_0 : n_0,\ q_1 : n_1,\ q_2 : n_2,\ q_3 : n_3,\ q_4 : n_4,\ q_5 : n_5,\ q_{12} : n_1 n_2,\ q_{34} : n_3 n_4,\ q_{14} : n_1 n_4,\ q_{23} : n_2 n_3\}$,
T $= \emptyset$,
$\Phi = \{a, b, c, g, t\}$,
$\Sigma = \{a, b, c, g, t, \lambda\}$,
γ: $\{[n_0, a, n_1], [n_0, b, n_2], [n_0, c, n_1 n_2], [n_1, g, n_3], [n_2, t, n_4], [n_3, a, n_5], [n_3 n_4, c, n_5], [n_4, b, n_5]\}$,
$\delta : \{[q_0, a, q_1], [q_0, b, q_2], [q_0, c, q_{12}], [q_1, g, q_3], [q_2, t, q_4], [q_{12}, t, q_{14}], [q_{12}, g, q_{23}], [q_{14}, g, q_{34}], [q_{23}, t, q_{34}], [q_3, a, q_5], [q_4, b, q_5], [q_{34}, c, q_5]\ \}$,
$q_0 = \{q_0\}$,
$F = \{q_5\}$ □

Como se ve en el ejemplo y se muestra en el patrón correspondiente, el proceso de multielección supone la separación de los posibles caminos, en conjuntos de estados disjuntos, para fusionarlos, sólo hay que añadir arcos desde los diferentes conjuntos de estados, a la secuencia de actividades final. Para ayudar a la clarificación del patrón, en el ejemplo se definen símbolos que definen la transición de la fusión, sin embargo, podrían ser cambiados a λ-transiciones o relojes c^0 para permitir su fusión inmediata.

Patrón 8: Multifusión

El patrón de multifusión es una alternativa funcional al patrón de fusión sincronizada. Mientras que el patrón de fusión sincronizada, exige que las secuencias de actividades que fueron seleccionadas inicialmente deben de terminar sincronizadamente, el patrón de multifusión permite a las secuencias fusionarse sin espera y continuar así el resto del proceso.

En la Figura A.8 se muestra un ejemplo de fusión multiple. En el ejemplo, la secuencia inicial es separada en hasta dos secuencias de actividades y a continuación son fusionadas

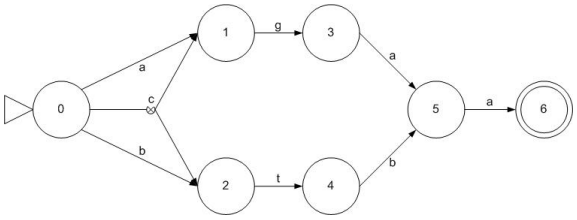

Figura A.9: Patrón de Discriminación

por separado sin arcos que exijan que los estados de los extremos finales de las secuencias (3 y 4)deban de estar activos. Una representación formal del ejemplo es definida a continuación:

A $=\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C $= \{c^\infty\}$,
P $= \{0,1,2,3,4,5,6\}$,
N $= \{n_0 : [0, c^\infty],\ n_1 : [1, c^\infty],\ n_2 : [2, c^\infty],\ n_3 : [3, c^\infty],\ n_4 : [4, c^\infty],\ n_5 : [5, c^\infty],\ n_6 : [6, c^\infty]\}$,
Q $= \{q_0 : n_0,\ q_1 : n_1,\ q_2 : n_2,\ q_3 : n_3,\ q_4 : n_4,\ q_5 : n_5,\ q_6 : n_6\ q_{12} : n_1n_2,\ q_{34} : n_3n_4,\ q_{14} : n_1n_4,\ q_{23} : n_2n_3,\ q_{15} : n_1n_5,\ q_{35} : n_3n_5,\ q_{25} : n_2n_5,\ q_{45} : n_4n_5,\ q_{16} : n_1n_6,\ q_{36} : n_3n_6,\ q_{26} : n_2n_6,\ q_{46} : n_4n_6,\ q_{56} : n_5n_6,\ q_{55} : n_5n_5,\ q_{66} : n_6n_6\ \}$,
T $= \emptyset$,
$\Phi = \{a, b, c, g, t\}$,
$\Sigma = \{a, b, c, g, t, \lambda\}$,
γ: $\{[n_0, a, n_1],\ [n_0, b, n_2],\ [n_0, c, n_1n_2],\ [n_1, g, n_3],\ [n_2, t, n_4],\ [n_3, a, n_5],\ [n_4, b, n_5],\ [n_5, a, n_6]\}$,
$\delta : \{[q_0, a, q_1],\ [q_0, b, q_2],\ [q_0, c, q_{12}],\ [q_1, g, q_3],\ [q_2, t, q_4],\ [q_{12}, t, q_{14}],\ [q_{12}, g, q_{23}],\ [q_{14}, g, q_{34}],\ [q_{25}, t, q_{45}],\ [q_{15}, g, q_{35}],\ [q_{35}, a, q_{56}],\ [q_{45}, b, q_{55}],\ [q_{56}, a, q_6],\ [q_{45}, a, q_{46}],\ [q_{15}, g, q_{35}],\ [q_{46}, b, q_5],\ [q_{26}, t, q_4],\ [q_{16}, g, q_3],\ [q_{36}, a, q_5],\ [q_{55}, a, q_{66}]\ \}$,
$q_0 = \{q_0\}$,
$F = \{q_6\}$ □

Como vemos en este ejemplo, la utilización de el patrón de fusión multiple requiere tener en cuenta todas las posibilidades de ejecución del proceso, teniendo en cuenta que las actividades que se ejecutan después del patrón de fusión multiple, pueden hacerlo más de una vez de una forma desfasada.

Patrón 9: Discriminador

Cuando se aplica un patrón de fusión multiple que ejecuta más de una secuencia de actividades a la vez, las actividades que se ejecutan después del patrón, lo hacen más de una vez y esto no es siempre deseable. La alternativa más obvia a este problema es la fusión sincronizada, sin embargo, no siempre es deseable a esperar a que se acaben las

actividades para continuar el proceso. De esta necesidad surge el patrón de discriminación. Este patrón, deja continuar la primera de las secuencias de actividades que lleguen a él ignorando el resto de las secuencias. De este modo se solucionan los problemas de la duplicidad de ejecución de actividades.

En la Figura A.9 se observa el ejemplo del patrón de discriminación. Aunque el gráfico es el mismo hay diferencias en su definición formal, como puede verse a continuación:

A $=\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C $= \{c^\infty\}$,
P $= \{0,1,2,3,4,5,6\}$,
N $= \{n_0 : [0, c^\infty],\ n_1 : [1, c^\infty],\ n_2 : [2, c^\infty],\ n_3 : [3, c^\infty],\ n_4 : [4, c^\infty],\ n_5 : [5, c^\infty],\ n_6 : [6, c^\infty]\}$,
Q $= \{q_0 : n_0,\ q_1 : n_1,\ q_2 : n_2,\ q_3 : n_3,\ q_4 : n_4,\ q_5 : n_5,\ q_6 : n_6\ q_{12} : n_1n_2,\ q_{34} : n_3n_4,\ q_{14} : n_1n_4,\ q_{23} : n_2n_3,\ q_{15} : n_1n_5,\ q_{35} : n_3n_5,\ q_{25} : n_2n_5,\ q_{45} : n_4n_5,\ q_{16} : n_1n_6,\ q_{36} : n_3n_6,\ q_{26} : n_2n_6,\ q_{46} : n_4n_6\}$,
T $= \emptyset$,
$\Phi = \{a, b, c, g, t\}$,
$\Sigma = \{a, b, c, g, t, \lambda\}$,
γ: $\{[n_0, a, n_1], [n_0, b, n_2], [n_0, c, n_1n_2], [n_1, g, n_3], [n_2, t, n_4], [n_3, a, n_5], [n_4, b, n_5], [n_5, a, n_6]\}$,
$\delta : \{[q_0, a, q_1], [q_0, b, q_2], [q_0, c, q_{12}], [q_1, g, q_3], [q_2, t, q_4], [q_{12}, t, q_{14}], [q_{12}, g, q_{23}], [q_{14}, g, q_{34}], [q_{23}, t, q_{34}], [q_{23}, a, q_{25}], [q_{14}, b, q_{15}], [q_{34}, b, q_{35}], [q_{34}, a, q_{45}], [q_{15}, g, q_{35}], [q_{25}, t, q_{45}], [q_{35}, a, q_6], [q_{45}, a, q_6], [q_{25}, a, q_{26}], [q_{15}, a, q_{16}], [q_{16}, g, q_{36}], [q_{26}, t, q_{46}]\}$,
$q_0 = \{q_0\}$,
$F = \{q_6\}$ □

En la descripción formal del ejemplo, mientras que en el patrón de fusión multiple se permitía la ejecución de todos los posibles caminos desde su aplicación, el patrón de discriminación no permite la continuación de las secuencias de actividades más allá de la ejecución del mismo.

A.3. Patrones Estructurales

Algunos sistemas de gestión de WF imponen restricciones en sus modelos de WF, como por ejemplo impedir los ciclos arbitrarios en las definiciones de WF. Estas restricciones no son siempre naturales, y tienden a restringir la libertad de especificación de los diseñadores de WF. En esta sección se presentan dos patrones que representan típicas restricciones estructurales de los sistemas de gestión de WF y que, no obstante, suelen ser interesantes para la definición de estos

Patrón 10: Ciclos Arbitrarios

Un ciclo arbitrario es un punto donde un conjunto de una o más actividades pueden ser ejecutadas repetidamente.

En la Figura A.10 podemos ver un patrón de ciclo arbitrario. un TPA no pone ninguna restricción a la utilización de arcos para la ejecución de procesos con lo que pueden ejecutarse complejos ciclos. La descripción formal del ejemplo puede verse a continuación.

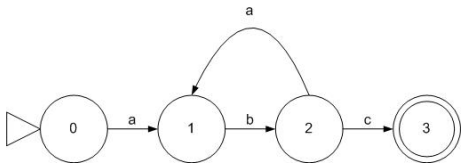

Figura A.10: Patrón de Ciclos Arbitrarios

A $=\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C $= \{c^{\infty}\}$,
P $= \{0,\ 1,\ 2,\ 3\}$,
N $= \{n_0 : [0, c^{\infty}],\ n_1 : [1, c^{\infty}],\ n_2 : [2, c^{\infty}],\ n_3 : [3, c^{\infty}]\}$,
Q $= \{q_0 : n_0,\ q_1 : n_1,\ q_2 : n_2,\ q_3 : n_3\}$,
T $= \emptyset$,
$\Phi = \{a, b, c\}$,
$\Sigma = \{a, b, c, \lambda\}$,
γ: $\{[n_0, a, n_1],\ [n_1, b, n_2],\ [n_2, a, n_1],\ [n_2, c, n_3]\}$,
δ: $\{[q_0, a, q_1],\ [q_1, b, q_2],\ [q_2, a, q_1],\ [q_2, c, q_3]\}$,
$q_0 = \{q_0\}$,
$F = \{q_3\}$ □

Como podemos ver, para ejecutar el cíclo sólo hemos tenido que crear un arco desde el punto final del ciclo al ínicio.

Patrón 11: Terminación Implicita

En algunos motores de WF un proceso sólo termina cuando se ha llegado a un estado final. Esto no es siempre deseable a nivel de implementación, ya que en determinados momentos, procesos que no llegan a estados finales deben terminar. El patrón de terminación implícita dice que un proceso debe terminar, no solo cuando llegue a un estado final, sino cuando ya no tenga ninguna actividad que hacer.

El problema de la terminación implícita es un problema a nivel de implementación de motor, por tanto no afecta a nivel del diseño de la aplicación y no tiene una representación formal asociada en un TPA.

A.4. Patrones que involucran Multiples instancias

Desde un punto de vista teórico, los patrones de multiples instancias corresponden a multiples hilos de ejecución que se refieren a las misma definición compartida. Sin embargo, desde un punto de vista practico, su implementación no es tan sencilla debido sobre todo a restricciones de diseño.

Cuando hablamos de patrones que involucran multiples instancias, consideramos dos tipos de habilidades: habilidades para lanzar multiples instancias de una actividad o subproceso, o habilidades para sincronizar estas actividades. Todos los patrones de esta sección cumplen el primer patrón, sin embargo, las diferencias entre los cuatro patrones de esta sección versan en la capacidad de la sincronización de éstas actividades o subprocesos.

En esta sección se definirán los patrones de multiples instancias, sin embargo, este es un problema a nivel de implementación del motor ya que la ejecución de las actividades de los estados depende de la actividad que el motor le asigne al estado en cuestión, por lo que no afecta al nivel de diseño y no afecta a las representaciones formales en TPAs.

Patrón 12: Multiples instancias Sin Sincronización

El patrón de multiples instancias sin sincronización define que una actividad es capaz de ejecutar instancias diferentes de la misma actividad (referenciada por un estado en un TPA). Estas instancias son independientes de las instancias de otros hilos, además de que no se requiere de sincronización entre ellas.

En un motor de TPA las nuevas instancias se ejecutarían conforme se fuese instanciando el estado.

Patrón 13: Multiples instancias con conocimiento en tiempo de Diseño

El patrón de multiples instancias con conocimiento en tiempo de diseño define que una actividad es capaz de ejecutar instancias diferentes de la misma actividad en un número conocido en tiempo de diseño del WF. Solo cuando todas las actividades se completasen las siguientes actividades se iniciarían.

En un motor de TPA el símbolo que ejecute el arco desde la actividad a sincronizar hasta el siguiente estado podría ser generado por el propio motor, cuando las actividades definidas hubiesen terminado.

Patrón 14: Multiples instancias con conocimiento en tiempo de Ejecución

El patrón de multiples instancias con conocimiento en tiempo de ejecución define que una actividad es capaz de ejecutar instancias diferentes de la misma actividad en un número conocido en tiempo de ejecución del WF. Solo cuando todas las actividades se completasen las siguientes actividades se iniciarían.

Del mismo modo que el caso anterior, en un motor de TPA el símbolo que ejecute el arco desde la actividad a sincronizar hasta el siguiente estado podría ser generado por el propio motor, cuando las actividades definidas en tiempo de ejecución hubiesen terminado.

Patrón 15: Multiples instancias sin conocimiento previo

El patrón de multiples instancias sin conocimiento previo define que una actividad es capaz de ejecutar instancias diferentes de la misma actividad en un numero desconocido

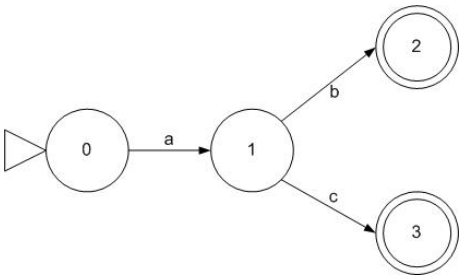

Figura A.11: Patrón de Elección derivada

a priori. Solo cuando todas las actividades se completasen las siguientes actividades se iniciarían.

Del mismo modo que los casos anteriores, el problema se puede resolver en un motor de TPA el símbolo que ejecute el arco desde la actividad a sincronizar hasta el siguiente estado podría ser generado por el propio motor, cuando las actividades ejecutadas hubiesen terminado.

A.5. Patrones Basados en Estados

En los flujos de trabajo de la vida real, las instancias normalmente se encuentran en un estado esperando a ser procesadas. Los patrones basados en estados son tres patrones en los que los procesos pueden estar esperando indefinidamente en un estado dependiendo de factores externos o internos al proceso.

En un TPA el concepto de estado de espera se produce justo al final de la ejecución de una actividad. En este momento, el proceso se encuentra esperando la llegada de un símbolo para ejecutar la siguiente actividad.

Patrón 16: Elección Derivada

El patrón de Elección Derivada define el proceso por el cual desde una actividad completada se pasa a un estado u otro dependiendo de la opción elegida. Este patrón se diferencia de los anteriores patrones de elección en que dicha elección la realiza un agente externo. El WF iniciado se queda en un estado suspendido hasta que el agente externo da la señal para la continuación del WF.

En la Figura A.11 se puede ver un ejemplo de patrón de elección derivada. En este gráfico se ve como se puede pasar del estado 1 al 2 con una b o del 1 al 3 con una c. Una representación formal se presenta a continuación:

A $=\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C $= \{c^{\infty}\}$,
P $= \{0, 1, 2, 3\}$,

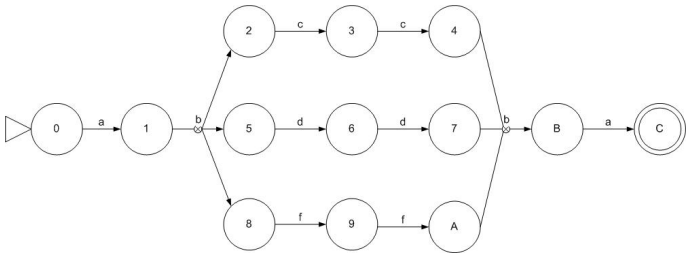

Figura A.12: Patrón de Rutina Paralela Entrelazada

N = $\{n_0 : [0, c^\infty]$, $n_1 : [1, c^\infty]$, $n_2 : [2, c^\infty]$, $n_3 : [3, c^\infty]\}$,
Q = $\{q_0 : n_0$, $q_1 : n_1$, $q_2 : n_2$, $q_3 : n_3\}$,
T = $\emptyset$,
$\Phi = \{a, b, c\}$,
$\Sigma = \{a, b, c, \lambda\}$,
γ: $\{[n_0, a, n_1]$, $[n_1, b, n_2]$, $[n_1, c, n_3]\}$,
δ: $\{[q_0, a, q_1]$, $[q_1, b, q_2]$, $[q_1, c, q_3]\}$,
$q_0 = \{q_0\}$,
$F = \{q_2, q_3\}$ □

Formalmente no existe diferencia entre este patrón y un simple patrón de elección. El WF puede quedarse esperando indefinidamente en el estado 1 hasta que un usuario externo, que puede interactuar a través del motor, genere el símbolo necesario para continuar.

Patrón 17: Rutina paralela entrelazada

El patrón de rutina paralela entrelazada define un proceso por el cual un conjunto de secuencias de actividades deben ejecutarse en un orden arbitrario, pero nunca dos actividades pueden ejecutarse al mismo tiempo. Si una de las secuencias de actividades empieza el resto han de permanecer en un estado de espera a que termine la primera para continuar.

En la Figura A.12 se puede ver un ejemplo de rutina paralela entrelazada. Existen tres secuencias de actividades correspondientes a los estados 2-3-4, 5-6-7 y 8-9-A que se han de ejecutar de manera arbitraria, pero sin ejecutarse al mismo tiempo. Los estados iniciales de las secuencias 2, 5 y 8 son estados de espera donde las actividades esperan a ser ejecutadas mientras, que los estados 4, 7 y A son estados finales de espera de cada una de las secuencias de actividades, donde una vez ejecutadas, estas esperan a que las demás secuencias terminen. Una representación formal se puede ver a continuación:

A =$\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C = $\{c^\infty\}$,

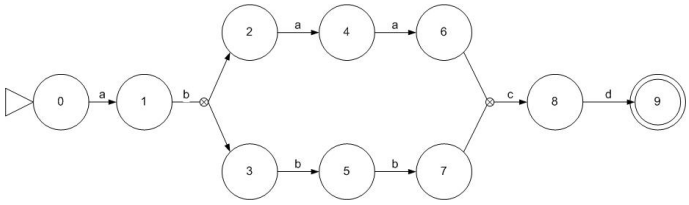

Figura A.13: Patrón de Hito

P = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C},
N = $\{n_0 : [0, c^\infty]$, $n_1 : [1, c^\infty]$, $n_2 : [2, c^\infty]$, $n_3 : [3, c^\infty]$, $n_4 : [4, c^\infty]$, $n_5 : [5, c^\infty]$, $n_6 : [6, c^\infty]$, $n_7 : [7, c^\infty]$, $n_8 : [8, c^\infty]$, $n_9 : [9, c^\infty]$, $n_A : [A, c^\infty]$, $n_B : [B, c^\infty]$, $n_C : [C, c^\infty]\}$,
Q = $\{q_0 : n_0$, $q_1 : n_1$, $q_{258} : n_2n_5n_8$, $q_{259} : n_2n_5n_9$, $q_{25A} : n_2n_5n_A$, $q_{268} : n_2n_6n_8$, $q_{26A} : n_2n_6n_A$, $q_{278} : n_2n_7n_8$, $q_{279} : n_2n_7n_9$, $q_{27A} : n_2n_7n_A$, $q_{358} : n_3n_5n_8$, $q_{35A} : n_3n_5n_A$, $q_{378} : n_3n_7n_8$, $q_{37A} : n_3n_7n_A$, $q_{459} : n_4n_5n_9$, $q_{45A} : n_4n_5n_A$, $q_{468} : n_4n_6n_8$, $q_{46A} : n_4n_6n_A$, $q_{478} : n_4n_7n_8$, $q_{479} : n_4n_7n_9$, $q_{47A} : n_4n_7n_A$, $q_B : n_B$, $q_C : n_C\}$,
T = $\emptyset$,
$\Phi = \{a, b, c, d, f\}$,
$\Sigma = \{a, b, c, d, f, \lambda\}$,
γ: $\{[n_0, a, n_1]$, $[n_1, b, n_2n_5n_8]$, $[n_2, c, n_3]$, $[n_3, c, n_4]$, $[n_5, d, n_6]$, $[n_6, d, n_7]$, $[n_8, f, n_9]$, $[n_9, f, n_A]$, $[n_4n_7n_A, b, n_B]$, $[n_B, a, n_C]\}$,
δ: $\{[q_0, a, q_1]$, $[q_1, b, q_{258}]$, $[q_{258}, c, q_{358}]$, $[q_{258}, d, q_{268}]$, $[q_{258}, f, q_{259}]$, $[q_{358}, c, q_{458}]$, $[q_{268}, d, q_{278}]$, $[q_{259}, f, q_{25A}]$, $[q_{458}, d, q_{468}]$, $[q_{458}, f, q_{459}]$, $[q_{468}, d, q_{478}]$, $[q_{459}, f, q_{45A}]$, $[q_{378}, c, q_{478}]$, $[q_{35A}, c, q_{45A}]$, $[q_{278}, c, q_{378}]$, $[q_{278}, f, q_{279}]$, $[q_{25A}, c, q_{35A}]$, $[q_{25A}, d, q_{26A}]$, $[q_{478}, f, q_{479}]$, $[q_{46A}, d, q_{47A}]$, $[q_{45A}, d, q_{46A}]$, $[q_{279}, f, q_{27A}]$, $[q_{26A}, d, q_{27A}]$, $[q_{479}, f, q_{47A}]$, $[q_{37A}, c, q_{47A}]$, $[q_{27A}, c, q_{37A}]$, $[q_{47A}, b, q_B]$, $[q_B, a, q_C]$ $\}$,
$q_0 = \{q_0\}$,
$F = \{q_C\}$ □

Como se puede ver, en la función de estados δ únicamente se contemplan las transiciones posibles, evitando así los estados prohibidos, que son aquellos en los que hay actividades solapadas. Una vez más, en el ejemplo todas las transiciones están etiquetadas, per si se hubiese querido que las transiciones fuesen inmediatas, se podrían utilizar λ-transiciones o relojes c^0.

Patrón 18: Hito

El patrón de Hito se basa en la idea en que la habilitación de una actividad de una secuencia de actividades depende de si un estado en una secuencia de actividades paralela ha sido alcanzado o no. En el caso en que el estado 'hito' no haya sido alcanzado, y la

secuencia no pueda continuar, la secuencia esperará indefinidamente hasta que el estado haya sido alcanzado.

En la Figura A.13 se puede observar un ejemplo de patrón de hito. En este ejemplo se toma que el estado 4 es un hito para la secuencia 3-5-7 y el estado 5 es un hito para la secuencia 2-4-6. Una especificación formal explicará mejor la solución:

A $=\{C, P, N, Q, T, \Phi, \Sigma, \gamma, q_0, F\}$
C $= \{c^\infty\}$,
P $= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, H\}$,
N $= \{n_0 : [0, c^\infty],\ n_1 : [1, c^\infty],\ n_2 : [2, c^\infty],\ n_3 : [3, c^\infty],\ n_4 : [4, c^\infty],\ n_5 : [5, c^\infty],\ n_6 : [6, c^\infty],\ n_7 : [7, c^\infty],\ n_8 : [8, c^\infty],\ n_9 : [9, c^\infty],\ n_H : [H, c^\infty]\}$,
Q $= \{q_0 : n_0,\ q_1 : n_1,\ q_{23} : n_2 n_3,\ q_{34} : n_3 n_4,\ q_{25} : n_2 n_5,\ q_{45} : n_4 n_5,\ q_{56} : n_5 n_6,\ q_{47} : n_4 n_7,\ q_{67} : n_6 n_7,\ q_8 : n_8,\ q_9 : n_9\ \}$,
T $= \emptyset$,
$\Phi = \{a, b, c, d\}$,
$\Sigma = \{a, b, c, d, \lambda\}$,
γ: $\{[n_0, a, n_1], [n_1, b, n_2 n_3], [n_2, a, n_4], [n_4, a, n_6], [n_3, b, n_5], [n_5, b, n_7], [n_6 n_7, c, n_8], [n_8, d, n_9]\}$,
δ: $\{[q_0, a, q_1], [q_1, b, q_{23}], [q_{23}, a, q_{34}], [q_{23}, b, q_{25}], [q_{34}, b, q_{45}], [q_{25}, a, q_{45}], [q_{45}, a, q_{56}], [q_{45}, b, q_{47}], [q_{47}, a, q_{67}], [q_{56}, b, q_{67}], [q_{67}, c, q_8], [q_8, d, q_9]\ \}$,
$q_0 = \{q_0\}$,
$F = \{q_9\}$ □

En la función δ del especificación formal se puede ver como el estado 45 resulta un hito. En esta especificación, sólo son válidos los arcos que definen el paso por los hitos 4 y 5. De forma análoga se podría utilizar solamente un estado de hito, solo que una de las secuencias no tendría restricciones para continuar.

A.6. Patrones de Cancelación

Los patrones de cancelación son referidos a la posibilidad de cancelar actividades o instancias en tiempo de ejecución.

Como en otros patrones, este tipo de patrones es totalmente dependiente del motor, con lo que no existe ninguna diferencia a nivel formal.

Patrón 19: Cancelar Actividad

El patrón de cancelar actividad se refiere al caso en el que la ejecución de una actividad es deshabilitada

Patrón 20: Cancelar Instancia

El patrón de cancelar Instancia se refiere al caso en el que la ejecución una instancia de un WF, es deshabilitado

Apéndice B

Workflows Inferidos en los Experimentos

En este anexo se presentan gráficamente los WF resultantes de los experimentos realizados.

Las Figuras B.1, B.2, B.3, B.4 y B.5 muestran los mejores WF resultantes de los algoritmos PALIA, HM, GPM, α y α++ respectivamente para el Experimento $Carepaths_A$

Con respecto al Experimento $Carepaths_B$, la Figura B.6 muestra el mejor resultado para PALIA, B.7 muestra el mejor de HM, B.8 hace lo propio para el algoritmo GPM, y B.9 y B.10 ilustran los mejores resultados para los algoritmos α y $\alpha++$

Por su parte el Experimento IC_A se representa en las Figuras: B.11 que representa el resultado de PALIA, B.12 el de HM, B.13 el de GPM, B.14 el de α y B.5 el de $\alpha++$.

Por ultimo el Experimento IC_B representa los mejores resultados de los algoritmos PALIA, HM, GPM, α y α++ en las Figuras B.16, B.17, B.18, B.19 y B.20 respectivamente.

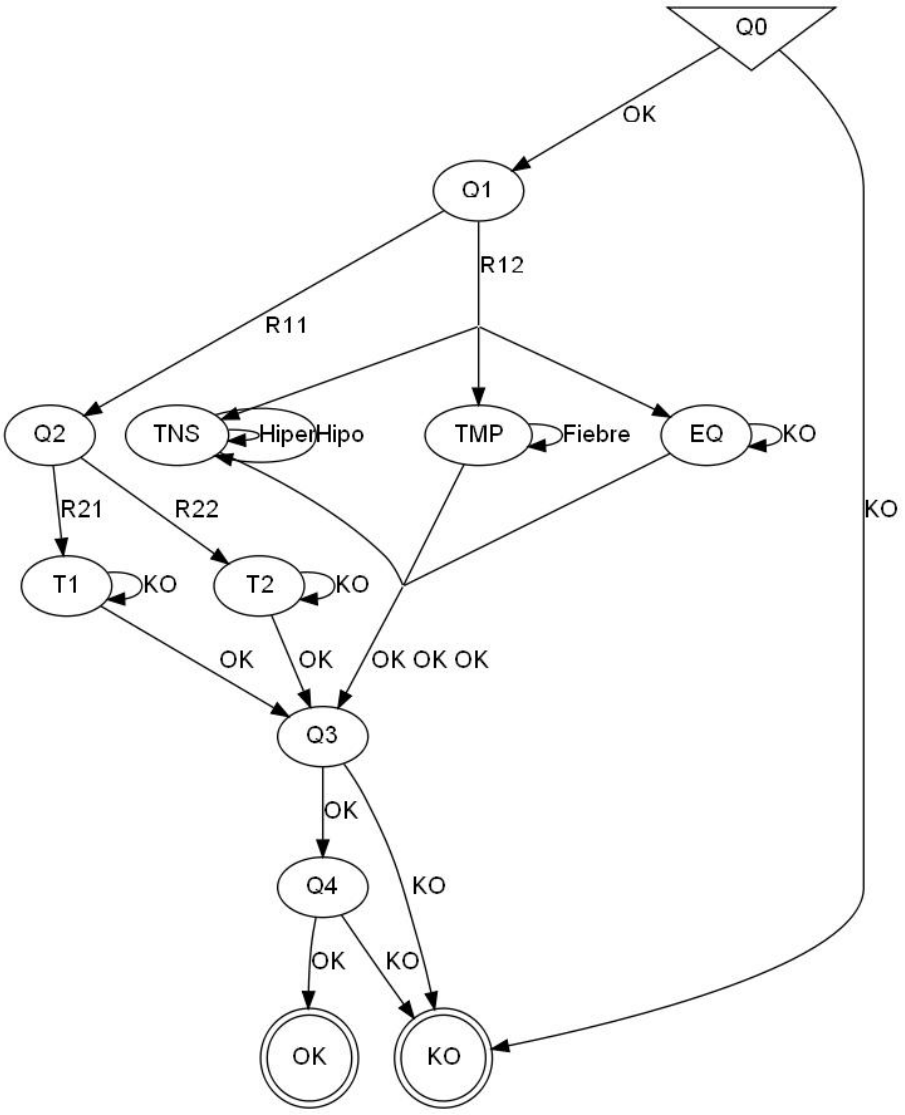

Figura B.1: Workflow del experimento $CarePaths_A$ inferido con PALIA

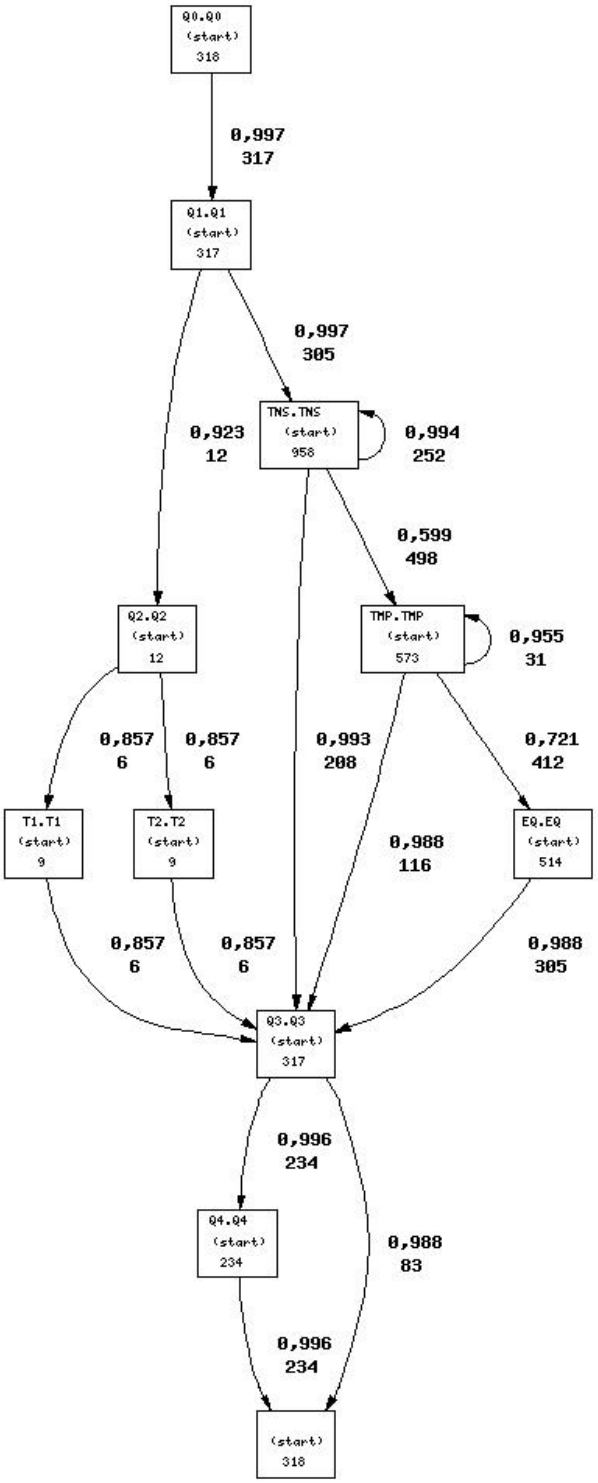

Figura B.2: Workflow del experimento $CarePaths_A$ inferido con Heuristic Miner Algorithm con información de inicio

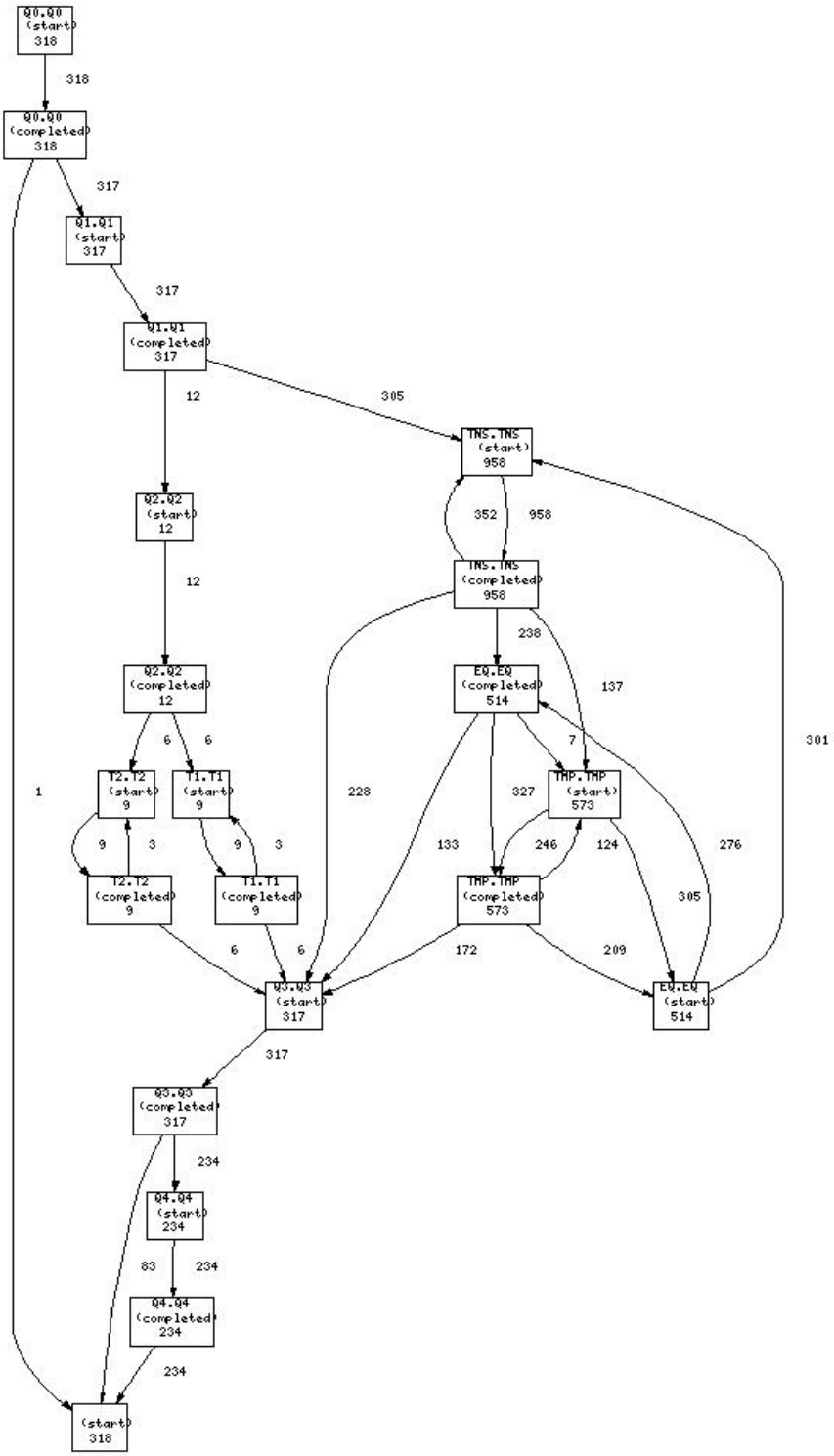

Figura B.3: Workflow del experimento $CarePaths_A$ inferido con Genetic Process Mining Algorithm con información inicial y final

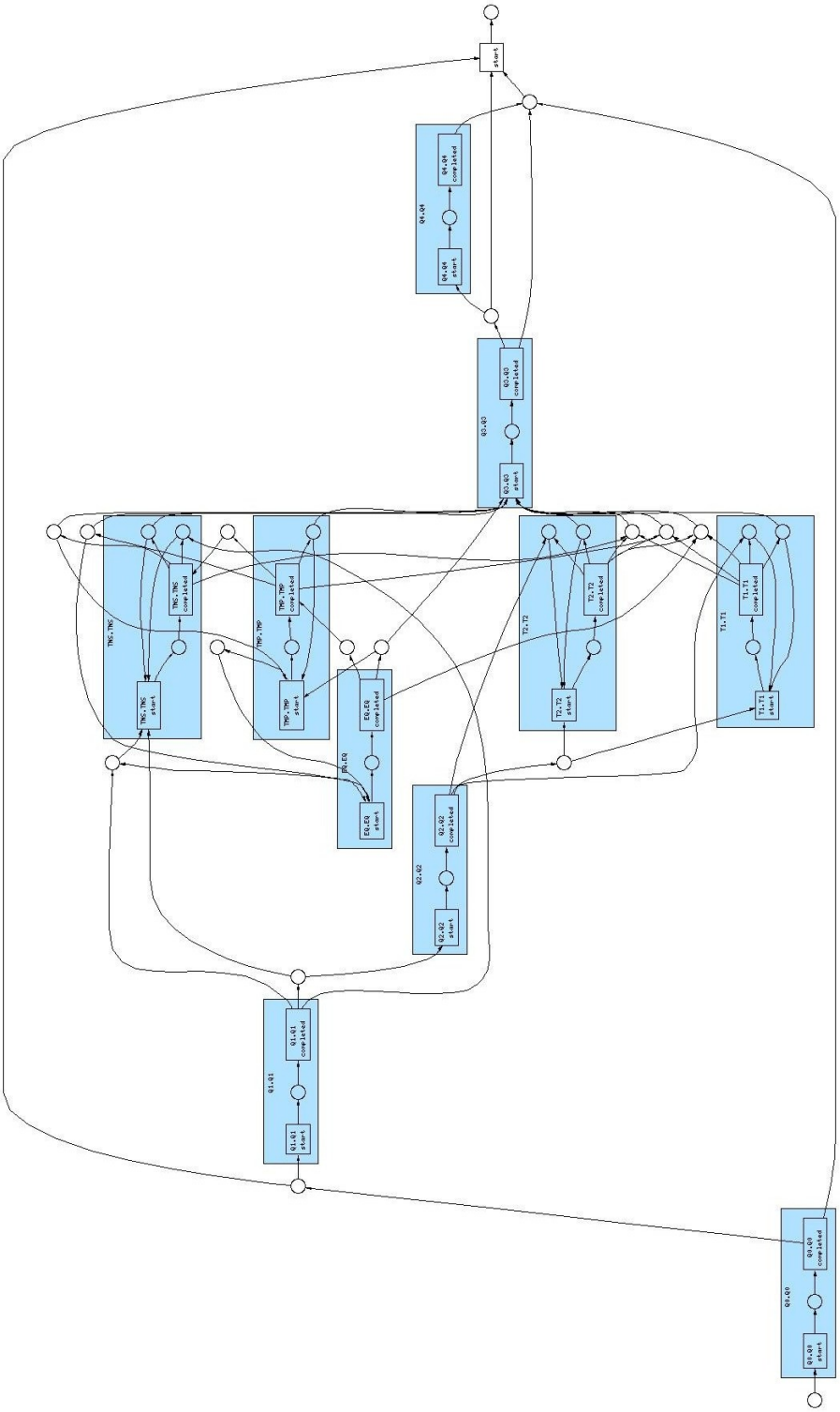

Figura B.4: Workflow del experimento $CarePaths_A$ inferido con α Algorithm utilizando información inicial y final

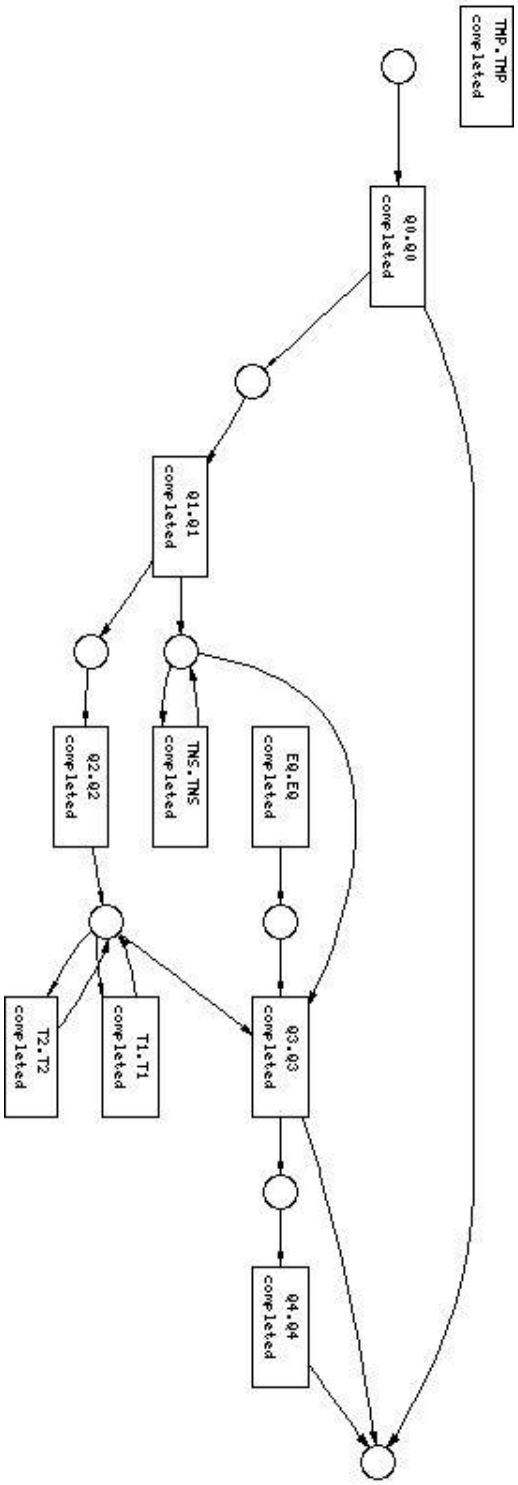

Figura B.5: Workflow del experimento $CarePaths_A$ inferido con α++ Algorithm utilizando información final

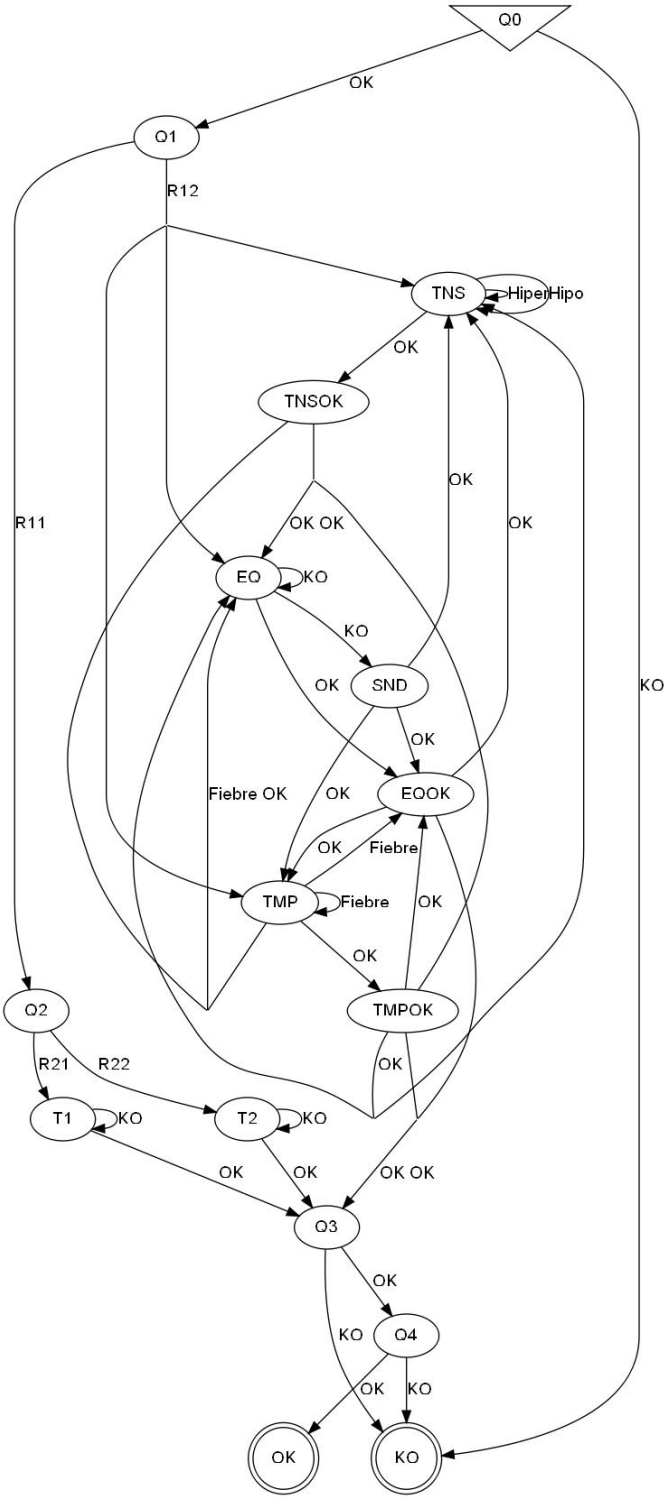

Figura B.6: Workflow del Experimento $CarePaths_B$ inferido con PALIA

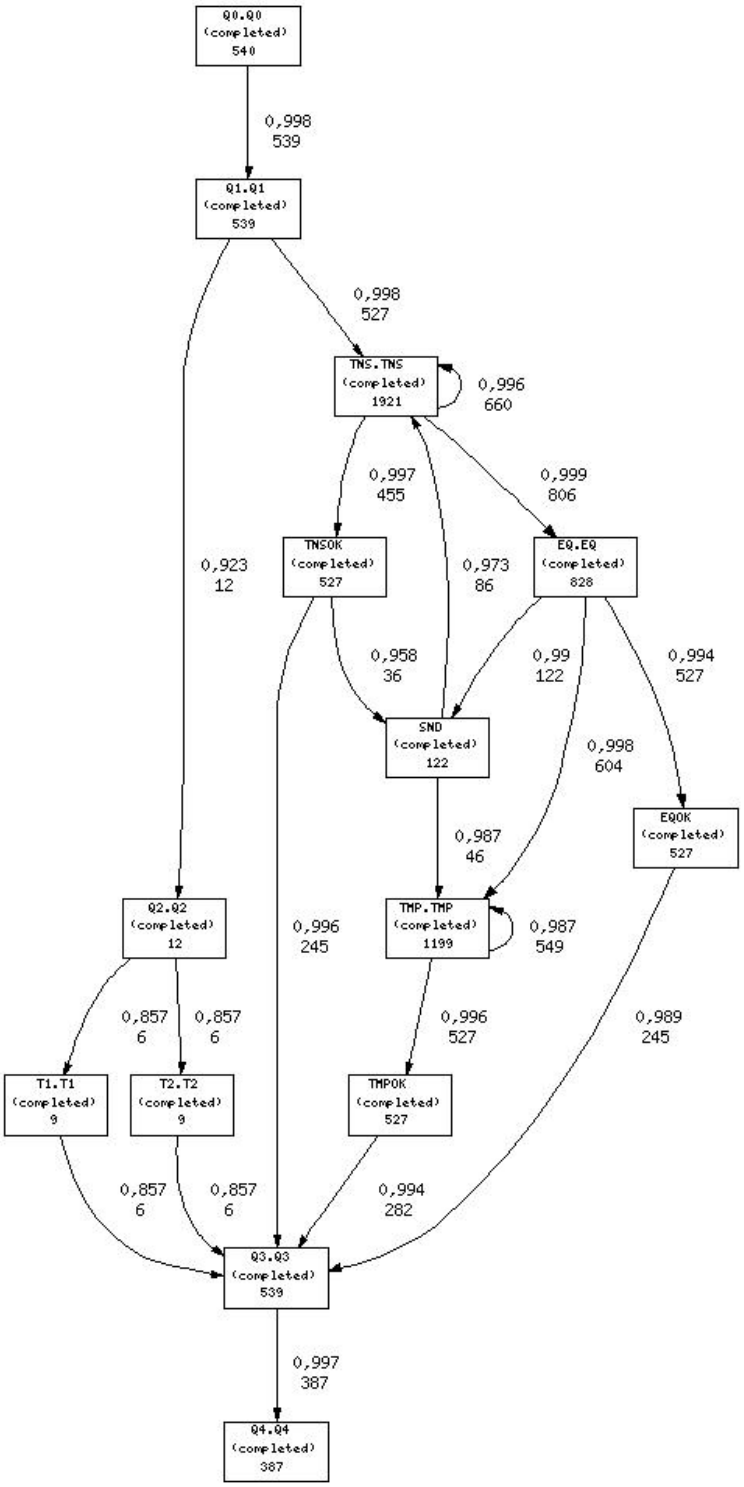

Figura B.7: Workflow del Experimento $CarePaths_B$ inferido con Heuristic Miner Algorithm con información final

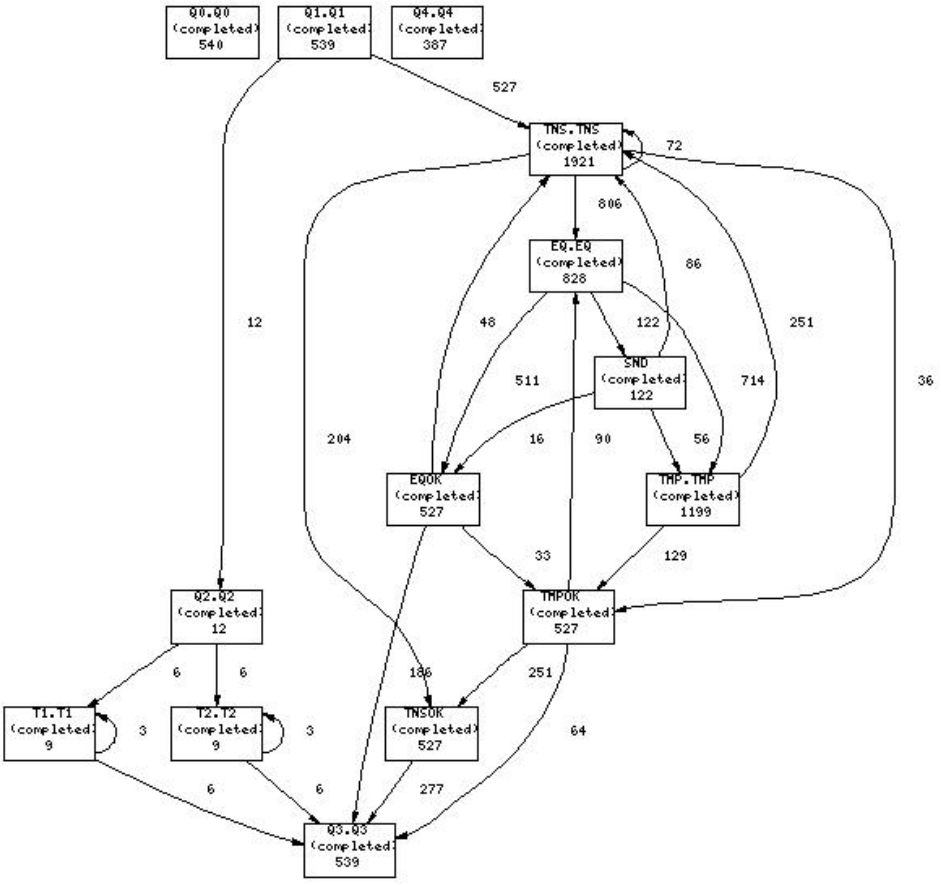

Figura B.8: Workflow del Experimento $CarePaths_B$ inferido con Genetic Process Mining Algorithm con información final

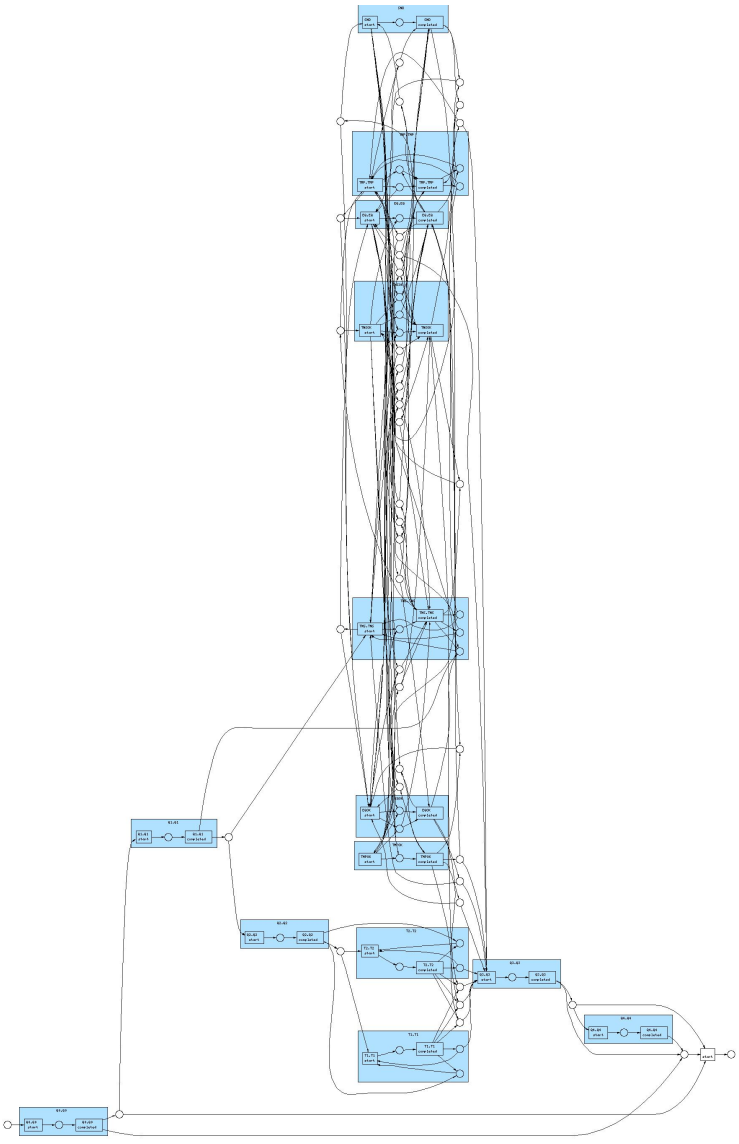

Figura B.9: Workflow del Experimento $CarePaths_B$ inferido con α Algorithm utilizando información inicial y final

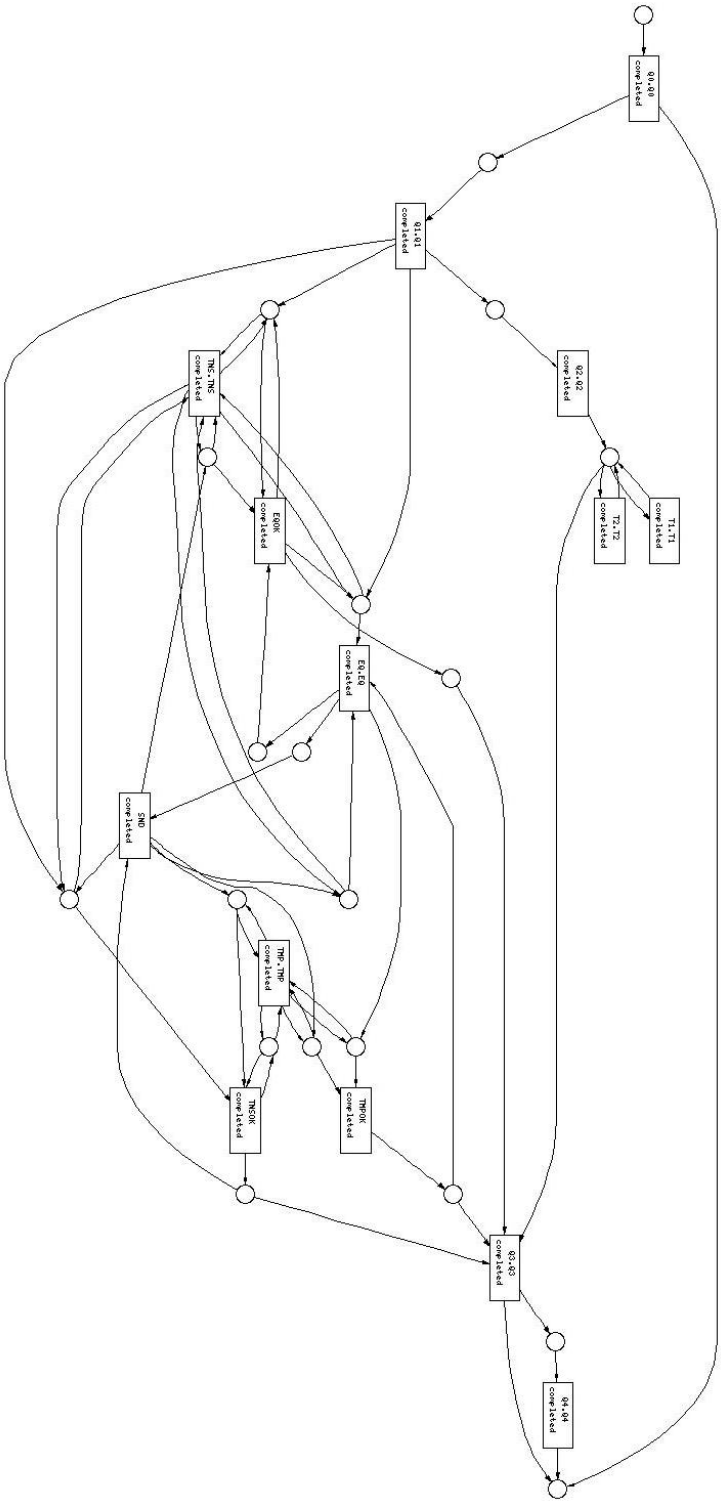

Figura B.10: Workflow del Experimento $CarePaths_B$ inferido con α++ Algorithm utilizando información final

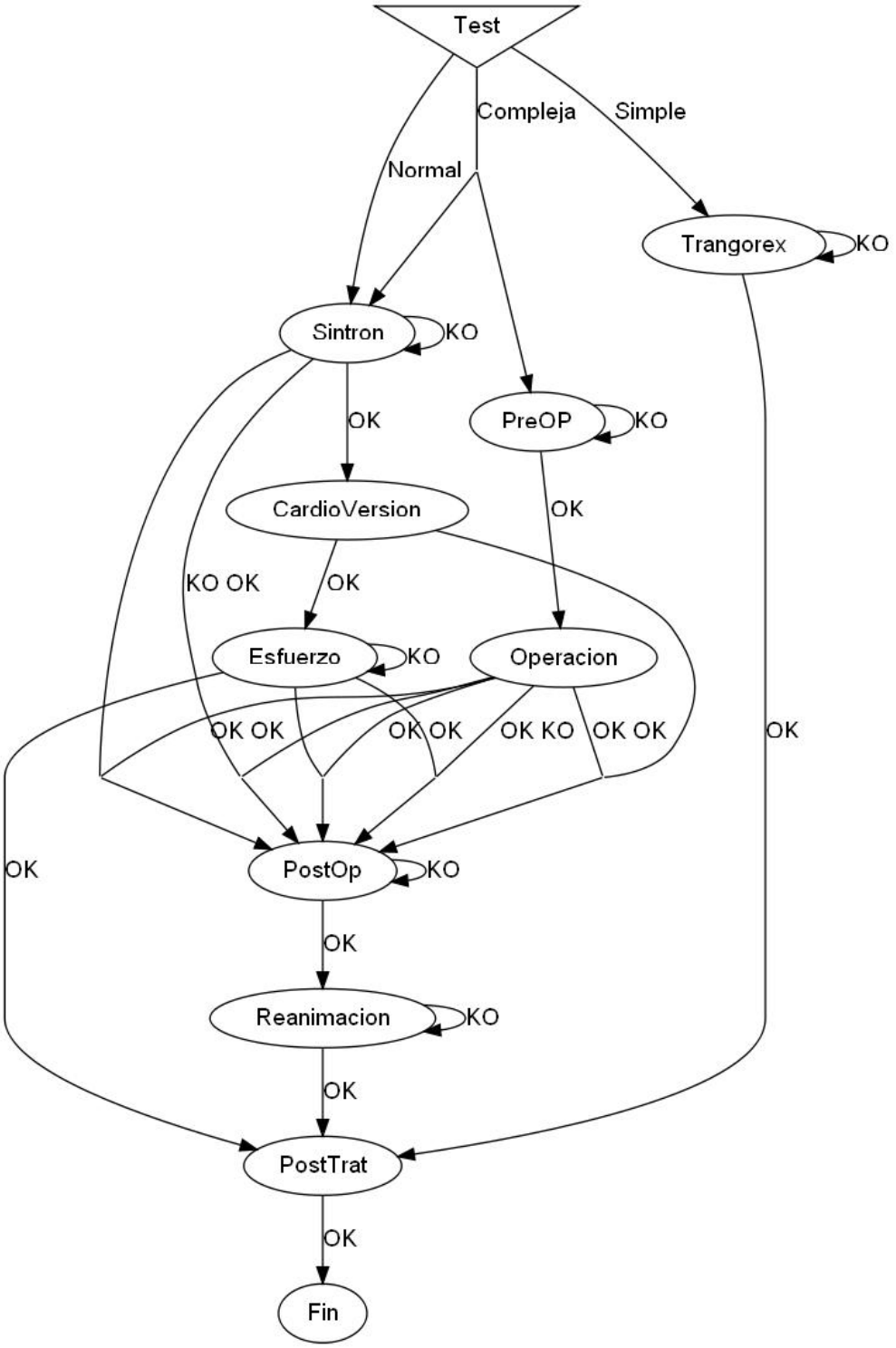

Figura B.11: Workflow del Experimento IC_A inferido con PALIA

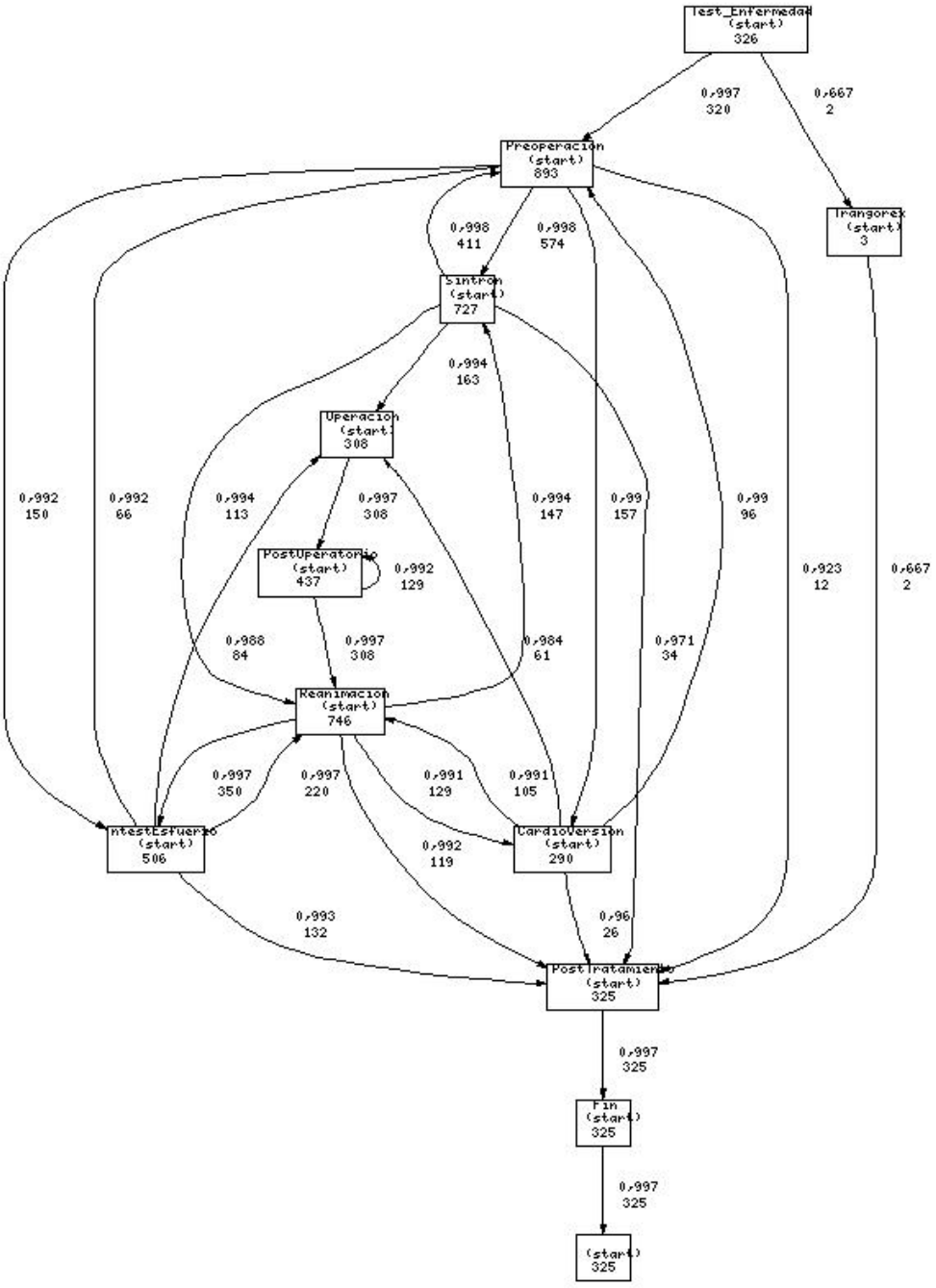

Figura B.12: Workflow del Experimento IC_A inferido con Heuristic Miner Algorithm con información inicial

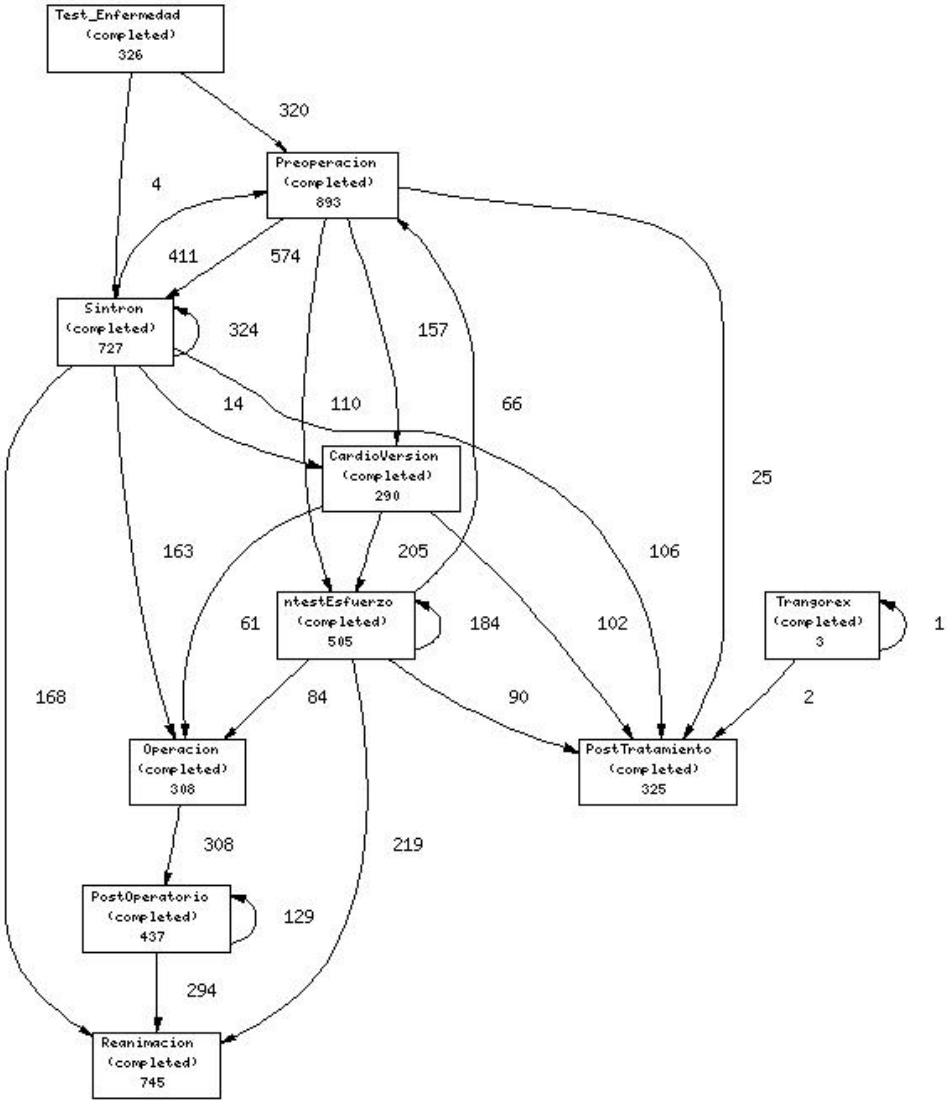

Figura B.13: Workflow del Experimento IC_A inferido con Genetic Process Mining Algorithm con información final

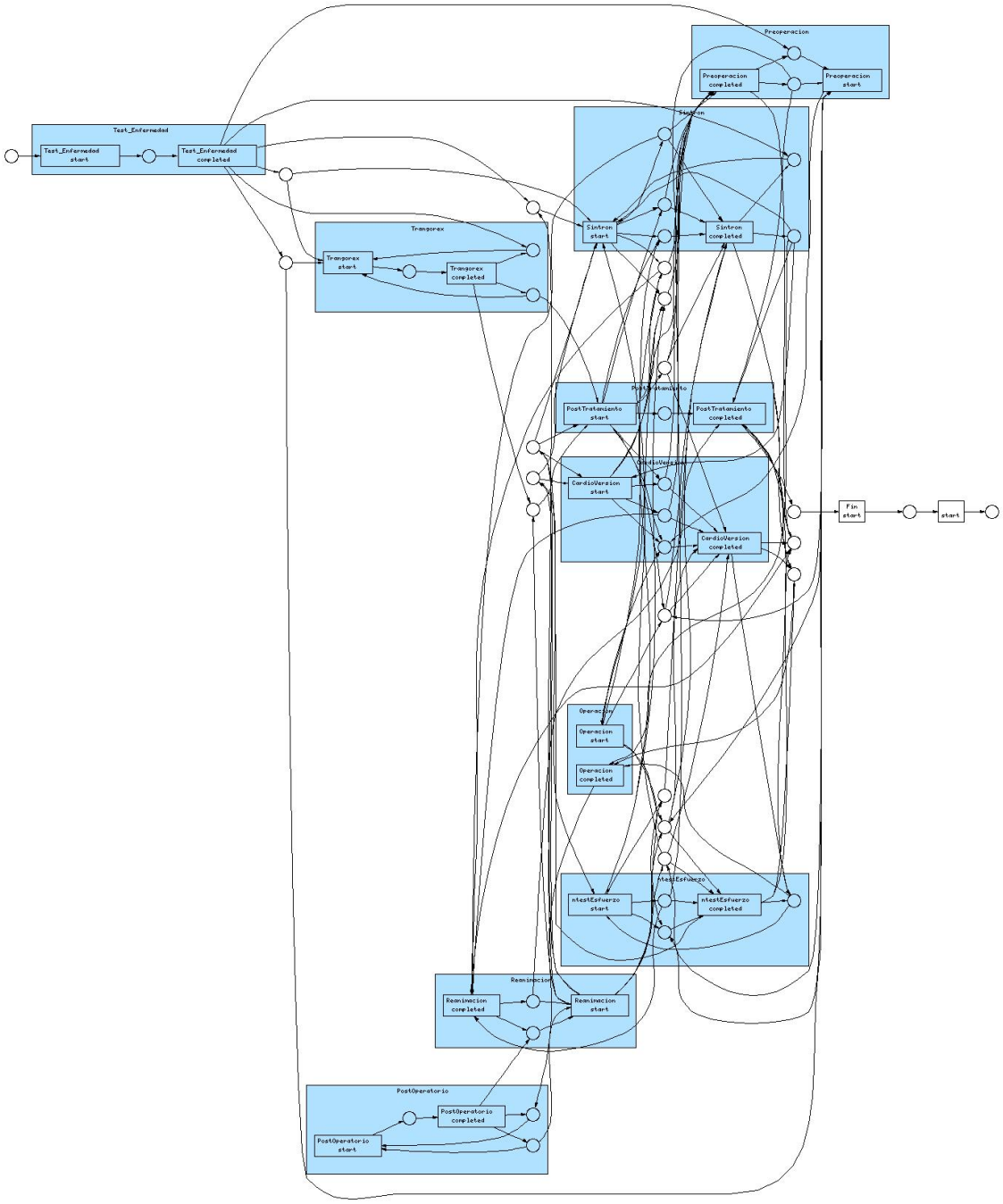

Figura B.14: Workflow del Experimento IC_A inferido con α Algorithm utilizando información inicial y final

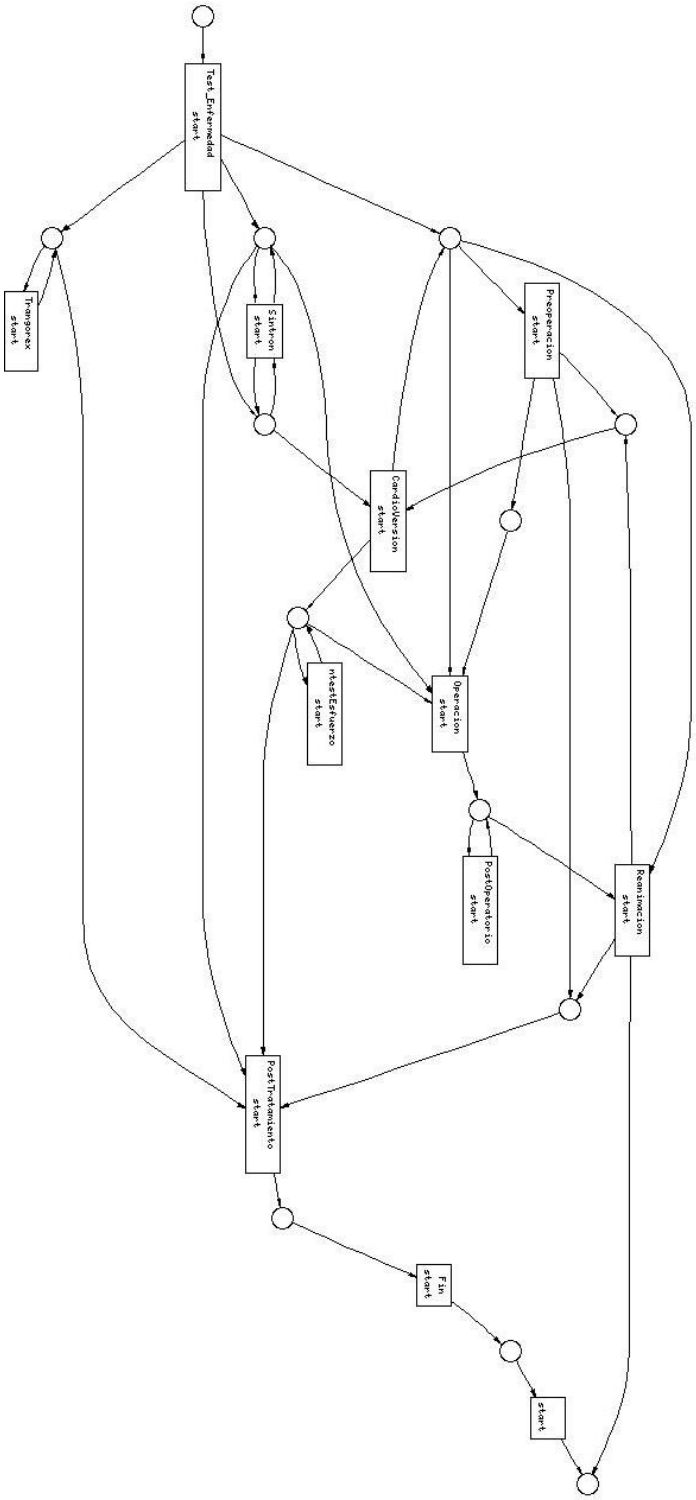

Figura B.15: Workflow del Experimento IC_A inferido con α++ Algorithm utilizando información Inicial

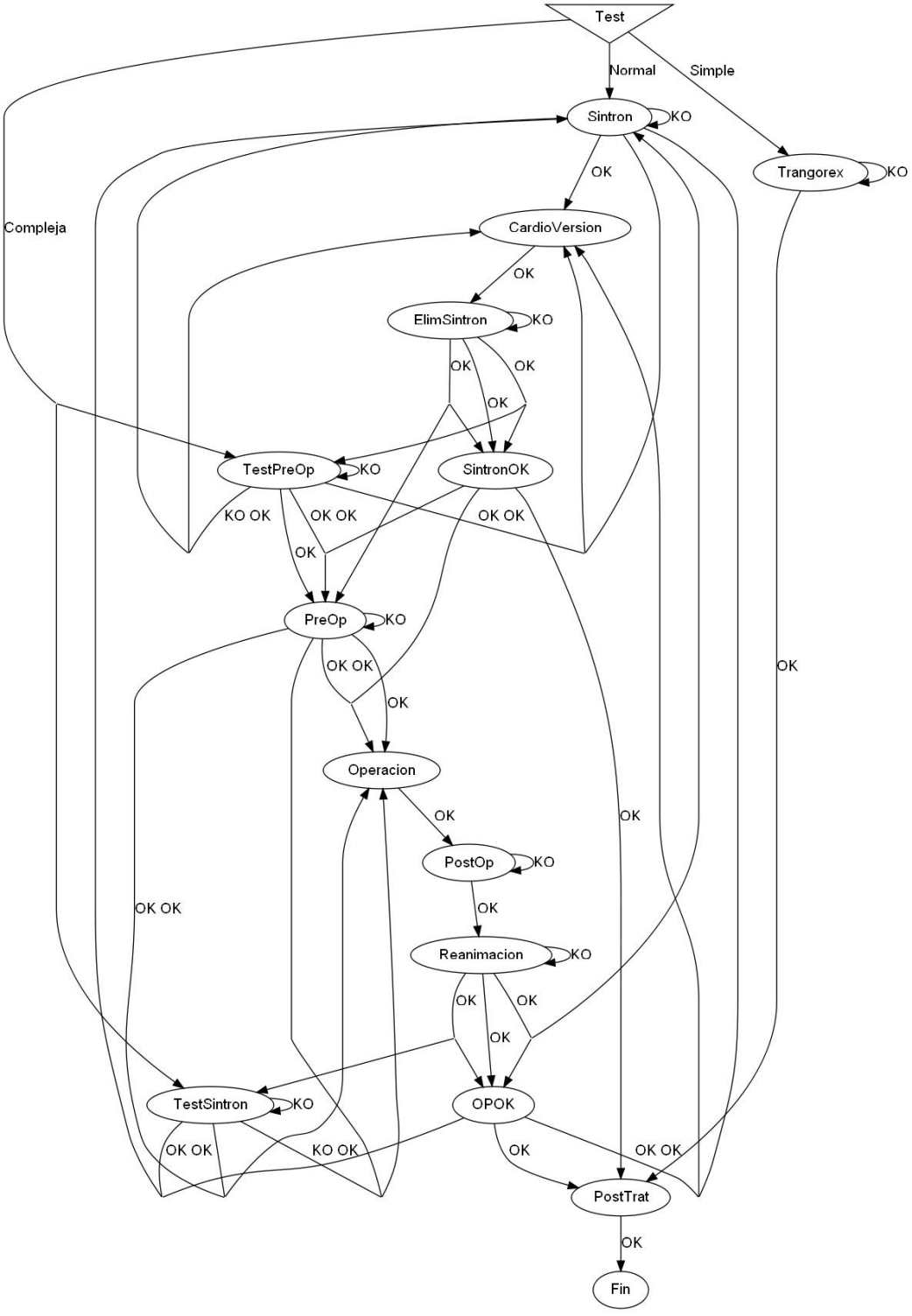

Figura B.16: Workflow del Experimento IC_B inferido con PALIA

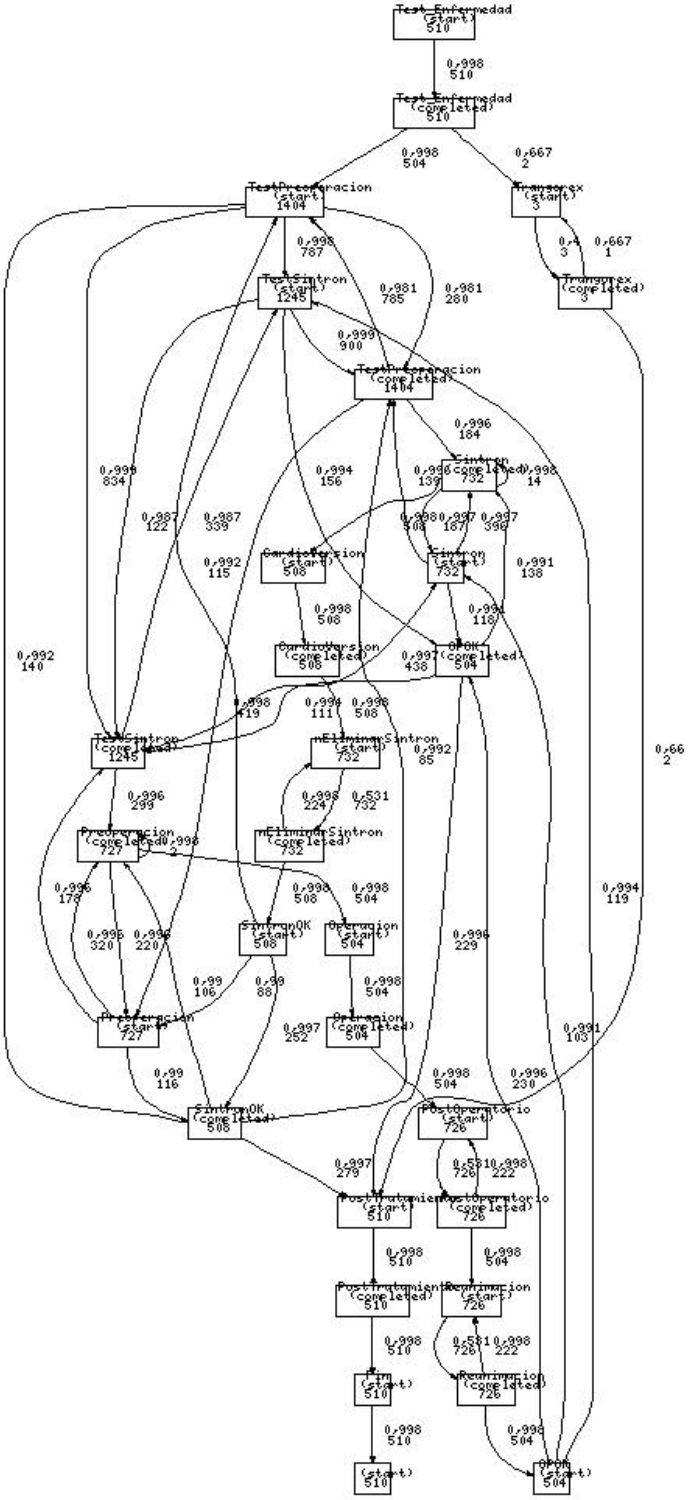

Figura B.17: Workflow del Experimento IC_B inferido con Heuristic Miner Algorithm con información inicial y final

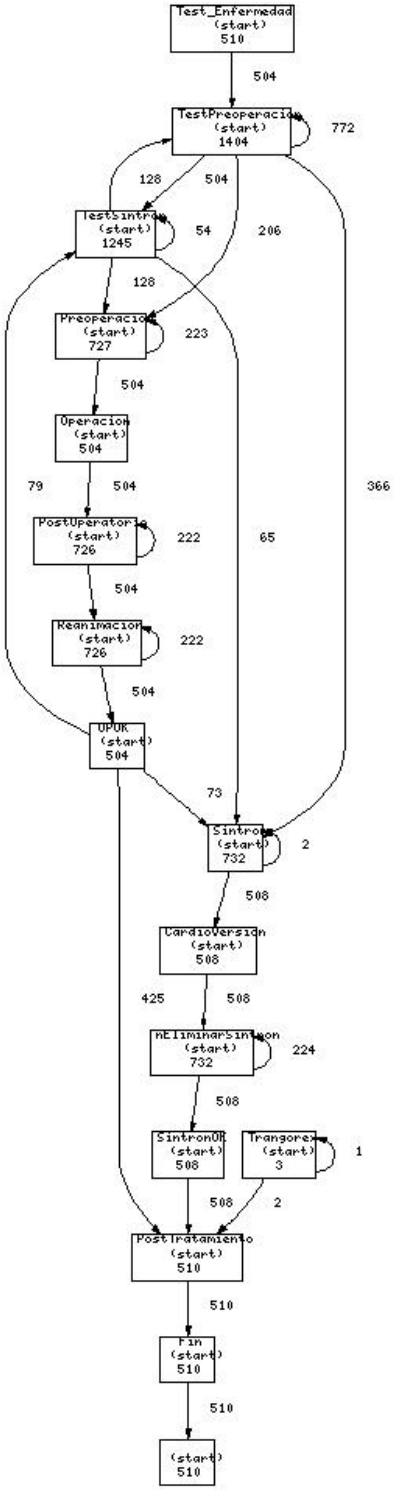

Figura B.18: Workflow del Experimento IC_B inferido con Genetic Process Mining Algorithm con información Inicial

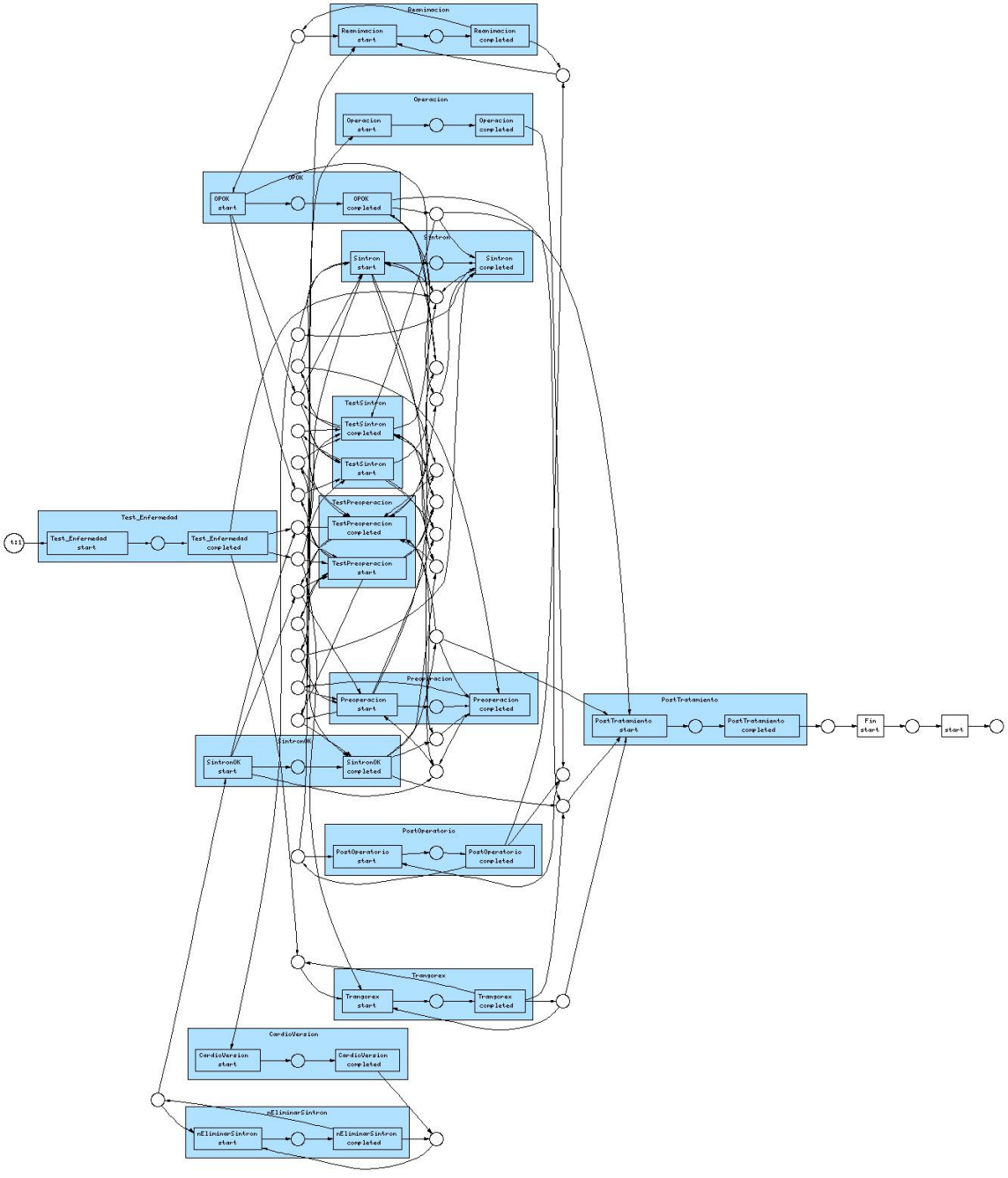

Figura B.19: Workflow del Experimento IC_B inferido con α Algorithm utilizando información inicial y final

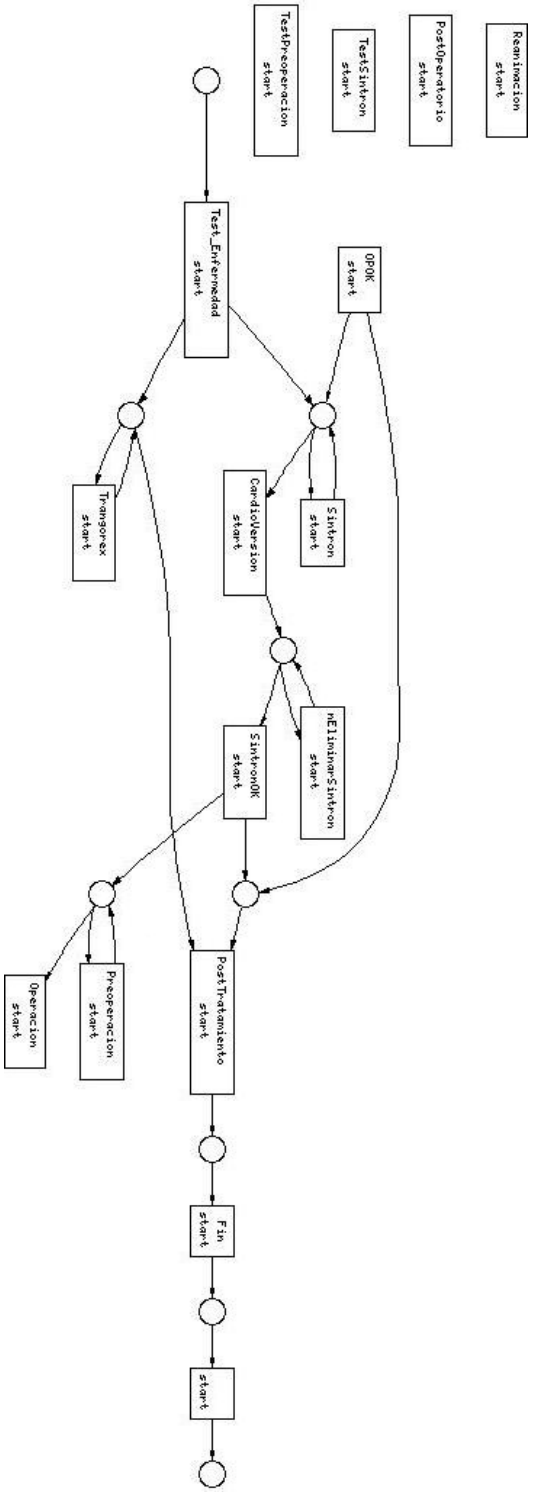

Figura B.20: Workflow del Experimento IC_B inferido con α++ Algorithm utilizando información inicial

Índice de figuras

Índice de tablas

Bibliografía

[1] VII Framework Program IST Project 045459. Persona project. perceptive spaces promoting independent aging.

[2] V Framework Program IST Project 20982. Agora 2000 project. innovative ist platforms and services to support a democratic regional/urban planning process, 2000-2003.

[3] VII Framework Program IST Project 216695. Heart cycle project:compliance and effectiveness in hf and chd closed-loop management, 2008-2011.

[4] VII Framework Program IST Project 224309. Vaalid project:accessibility and usability validation framework for aal interaction design process, 2008-2011.

[5] V Framework Program IST Project 34614. Ideas in e-health project. integrated distributed environment for application services in e-health.

[6] VI Framework Program IST Project 507017. Carepaths project. an intelligent support environment to improve the quality of decision processes in health communities, 2004-2007.

[7] VI Framework Program IST Project 507019. Pips project. personalised information platform for life and health services.

[8] VI Framework Program IST Project 507816. Myheart project. fighting cardiovascular diseases by prevention and early diagnosis, 2003-2008.

[9] Jr. Adilson Arcoverde, Jr. Gabriel Alves, and Ricardo Lima. Petri nets tools integration through eclipse. In *eclipse05: Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*, pages 90–94, New York, NY, USA, 2005. ACM.

[10] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1988.

[11] Rajeev Alur and David L. Dill. A theory of timed automata. Technical report, Computer Science Department Stanford University, 1994.

[12] Rajeev Alur and P. Madhusudan. Decision problems for timed automata: A survey. Technical report, University of Pensylvania, 2004.

[13] Scott Ambler. *Agile Model Driven Development with UML 2.0*. Cambridge University Press, 2004.

[14] S. Audimoolan, M.Ñair, R. Gaikward, and C. Qing. The role of clinical pathways. *Improving Patient Outcomes*, February:–, 2005.

[15] Patricia Bouyer. Weighted timed automata: model-checking and games. In *Proceedings of the Pre-FSTTCS 2006 Workshop Kolkata, India*, 2006.

[16] **Carlos Fernandez** and Jose Miguel Benedí. Timed parallel automaton learning in workflow mining problems. In *Ciencia y Tecnología en la Frontera ISSN:1665-9775*, 2008.

[17] **Carlos Fernandez** and Jose Miguel Benedi. Activity-based workflow mining: A theoretical framework. In *Workshop On Technologies for Healthcare and Healthy Lifestyle ISBN:978-84-611-1311-8*, 2006.

[18] **Carlos Fernandez**, Juan Pablo Lazaro, and Jose Miguel Benedi. Workflow mining application to ambient intelligence behaviour modelling. In *Human Computer Interaction International 2009. Aceptado, Pendiente de publicación*, 2009.

[19] **Carlos Fernandez**, Juan Pablo Lazaro, Gema Ibañez, and David Dominguez. Workflow mining para el modelado individualizado del comportamiento humano en el contexto de la inteligencia ambiental. In *Ciencia y Tecnología en la Frontera ISSN:1665-9775*, 2008.

[20] **Carlos Fernandez**, Juan Carlos Naranjo, Eduardo Monton, Vicente Traver, and Bernardo Valdivieso. Gestion de hospitalizacion a domiclio distribuida mediante servicios web. In *Congreso Anual de la Sociedad Española de Ingeniería Biomedica (CASEIB)*, 2003.

[21] **Carlos Fernandez**, Carlos Sanchez, Vicente Traver, and Jose Miguel Benedí. Tpaengine: Un motor de workflows basado en tpas. In *Ciencia y Tecnología en la Frontera ISSN:1665-9775*, 2008.

[22] David Chappell. *Enterprise Service Bus*. OReilly, 2004.

[23] Jorge Civera, Juan Miguel Vilar, Elsa Cubel, Antonio L. Lagarda, Sergio Barrachina, Francisco Casacuberta, and Enrique Vidal. A novel approach to computer-assisted translation based on finite-state transducers. In *FSMNLP*, pages 32–42, 2005.

[24] Workflow Management Coalition. *Process Definition Interface – XML Process Definition Language*. WFMC-TC-1025, Document Status Final, 2005.

[25] The Cochrane Collaboration. Cochrane library: http://www.cochrane.org/index.htm.

[26] CarePaths Consortium. Deliverable d3.2 pathways design, management and evaluation tools. Technical report, CarePaths Project, 2005.

[27] Heart Cycle Consortium. Deliverable d4.5 preliminary reference architecture for personalized healthcare of the chronic condition. Technical report, Heart Cycle Project, 2008.

[28] MyHeart Consortium. Deliverable d2 concept realization plan state of the art in technology. Technical report, MyHeart Project, 2004.

[29] MyHeart Consortium. Deliverable d7 pechnical concept manifestation. Technical report, MyHeart Project, 2004.

[30] PERSONA Consortium. Deliberable d3.3.1 reference architecture and information model for service infrastructureă parte b orchestration services. Technical report, PERSONA Project, 2008.

[31] PIPS Consortium. Deliverable d4.2.3 lap support engine implementation. Technical report, PIPS Project, 2007.

[32] World Wide Web Consortium. W3c web services standard: http://www.w3.org/2002/ws/.

[33] Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.

[34] Microsoft Corporation. Introducing microsoft windows workflow foundation: http://msdn2.microsoft.com/en-us/library/aa480215.aspx.

[35] Francois Coste. State merging inference of finite state classifiers. In *Rapport technique n INRIA/RR-3695, IRISA, septembre*, 1999.

[36] Pedro R. D'Argenio. Regular processes and timed automata. In *Proceedings of Fourth AMAST Workshop on Real-Time Systems, Concurrent, and Distributed Software (ARTS)*, pages 141–155, 1997.

[37] T. Davenport. The coming commoditization of processes. *Harvard Business Review*, 2005.

[38] T. Davenport and J. Short. The new industrial engineering: Information technology and business process redesign. *Sloan Management Review*, pages 11–27, 1990.

[39] Thomas Davenport. *Process Innovation: Reengineering work through information technology.* Harvard Business School Press, 1993.

[40] A. K. Alves de Medeiros, B. F. Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters. Process mining extending the alpha algorithm to mine short loops. Technical report, WP113 Beta Paper Series Eindhoven University of Technology, 2004.

[41] Ana Karla A. de Medeiros, A. J. M. M. Weijters, and W.M.P. van der Aalst. Genetic process mining: A basic approach and its challenges. In *Business Process Management Workshops*, pages 203–215, 2005.

[42] Proyecto Alcoy Ciudad Digital. http://www.alcoy.es/isum/.

[43] JBoss Red Hat Division. Jboss jbpm white paper: http://jboss.com/pdf/jbpm whitepaper.pdf.

[44] David Dominguez, **Carlos Fernandez**, Teresa Meneu, Juan Bautista Mocholi, and Riccardo Seraffin. Medical guidelines for the patient: Introducing the life assistance protocols. In *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*, volume 139, page 282. IOS Press, 2008.

[45] B. F. Van Dongen. Emit software and test corpus: http://prom.win.tue.nl/research/wiki/tools.

[46] B. F. Van Dongen. Emit: A process mining tool. In *Application and Theory of Petri Nets 2004, volume 3099 of Lecture Notes in Computer Science*, pages 454–463. Springer-Verlag, 2004.

[47] A.S. Elstein. On the origins and development of evidence-based medicine and medical decision making. *Springer*, ago 2004:184–189, 2004.

[48] Miguel Valdes et all. Bonita site: http://wiki.bonita.objectweb.org/xwiki/bin/view/main/.

[49] James Evans and James Dean. *Total Quality: Management, Organization and Strategy*. South-Western Pub, 2007.

[50] N.R. Every, J. Hochman, R. Becker, S. Kopecky, and C.P. Cannon. Critical pathways. a review. *Circulation*, 101:461–465, 2000.

[51] M. Field and K. Lohr. *Clinical practice guidelines: directions for a new program.* National Academy Press, 3 edition, 1990.

[52] Fisterra. Biblioteca de vias clínicas fisterra: http://www.fisterra.com/fisterrae/.

[53] Organization for the Advancement of Structured Information Standards. Oasis wsbpel standard: http://docs.oasis-open.org/wsbpel/2.0/os/wsbpel-v2.0-os.html.

[54] Walid Gaaloul, Sadek Alaoui, Karim Baina, and Claude Godart. Mining workflow patterns through event-data analysis. In *SAINT-W '05: Proceedings of the 2005 Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops)*, pages 226–229, Washington, DC, USA, 2005. IEEE Computer Society.

[55] Pedro García. *Explorabilidad Local e Inferencia Inductiva de Lenguajes Regulares y Aplicaciones*. PhD thesis, DSIC. Universidad Politécnica de Valencia, 1988.

[56] Robert Geist, Darren Crane, Stephen Daniel, and Darrell Suggs. Systems modeling with xpetri. In *WSC94: Proceedings of the 26th conference on Winter simulation*, pages 611–618, San Diego, CA, USA, 1994. Society for Computer Simulation International.

[57] B. Grahlmann, M. Moeller, and U. Anhalt. A new interface for the pep tool – parallel finite automata –. In *2. Workshop Algorithmen und Werkzeuge für Petrinetze*, 22, pages 21–26, 1995.

[58] Bernd Grahlmann. Combining finite automata, parallel programs and sdl using petri nets. In Bernhard Steffen, editor, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1384 of *Lecture Notes in Computer Science*, pages 102–117. Springer-Verlag, 1998.

[59] Object Management Group. *Business Process Modeling Notation (BPMN) Specification.* OMC dtc/06-02-01, 2006.

[60] Sergio Guillen, Teresa Arredondo, Vicente Traver, Juan Miguel García, and **Carlos Fernandez**. Multimedia telehomecare system using standard tvset. *IEEE Transactions on Biomédical Engineering ISSN:0276-6547 22 Citas JCR:1.665*, 49:1431–1437, 2002.

[61] Seungchul Ha and Hyo-Won Suh. A timed colored petri nets modeling for dynamic workflow in product development process. *Comput. Ind.*, 59(2-3):193–209, 2008.

[62] Michael Hammer. Reengineering work: Do not automate, obliterate. *Hardvard Business Review*, jul/ago 1990:104–112, 1990.

[63] Peter J. Hass. Stochastic petri nets for modelling and simulation. In J.S. Smith R.G. Ingalls, M.D. Rossetti and B.A. Peter Eds, editors, *Proceedings of 2004 Winter Simulation Conference*, 2004.

[64] TIBCO Software Inc. Staffware processsuite whitepaper: http://about.reuters.com/partnerships/tibco/material/staffware whitepaper.pdf.

[65] Petr Jancar, Javier Esparza, and Faron Moller. Petri nets and regular processes. *Journal of Computer and System Sciences*, 59(3):476–503, 1999.

[66] Kurt Jensen. An introduction to the practical use of coloured petri nets. In *Petri Nets (2)*, pages 237–292, 1996.

[67] G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation.* Addison Wesley, 1998.

[68] Thor Kristoffersen, Anders Moen, and Halltein Asheim Hansen. Extracting high-level information from petri nets: A railroad case. In Estonian Academy of Physics and Mathematics, editors, *Proceedings of the Estonian Academy of Physics and Mathematics*, 2003.

[69] Robert Kristofl and Thomas Grechenig. Applying workflow management concepts in public administration: a case study in the austrian ministry of science and research. In *CON '94: Proceedings of the ninth Austrian-informatics conference on Workflow management : challenges, paradigms and products*, pages 201–211, Munich, Germany, Germany, 1994. R. Oldenbourg Verlag GmbH.

[70] H.G. Mendelbaum and R.B. Yehezkael. Using parallel automaton as a single notation to specify, design and control small computer based systems. In *IEEE 2001 -ECBS*, 2001.

[71] Juan Bautista Mocholi, David Dominguez, and **Carlos Fernandez**. Towards an environment under which executing laps. In *Workshop On Technologies for Healthcare and Healthy Lifestyle ISBN:978-84-611-1311-8*, 2006.

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

[73] Juan Carlos Naranjo, **Carlos Fernandez**, Susana Pomes, and Bernardo Valdivieso. Care-paths: Searching the way to implement pathways. *Computers in Cardiology ISSN:0276-6547*, 33:285–288, 2006.

[74] Juan Carlos Naranjo, **Carlos Fernandez**, Maria Pilar Sala, Juan Bautista Mocholi, Michael Hellenschmidt, and Franco Mercalli. A modelling framework for ambient assisted living validation. In *Human Computer Interaction International 2009. Aceptado, Pendiente de publicación*, 2009.

[75] Juan Carlos Naranjo, **Carlos Fernandez**, Salvador Vera, Sergio Guillen, and Bernardo Valdivieso. Proyecto ideas: Systema de informacion distribuido para una unidad de hospitalización domiciliaria. In *Congreso Nacional de Informatica para la Salud(INFORSALUD)*, 2002.

[76] National Library of Medicine and The National Institutes of Health. Pubmed library: http://www.pubmed.gov.

[77] University of Technology Eindhoven and University of Technology Queensland. Workflow patterns initiative: http://www.workflowpatterns.com/.

[78] Jose Oncina and Pedro García. *Inferring regular languages in polynomial update time*. World Scientific Publishing, 1991. Pattern Recognition and Image Analysis 1991.

[79] Jose Oncina, Pedro Garcia, and Enrique Vidal. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 448–458, 1993.

[80] H. Oswald, R. Esser, and R. Mattmann. An environment for specifying an executing hierarchical petri nets. In *ICSE '90: Proceedings of the 12th international conference on Software engineering*, pages 164–172, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[81] David Platt. *Introducing the Microsoft .NET framework*. Microsoft Press, 2001.

[82] Roger S. Pressman. *Ingeniería del Software: Un enfoque práctico*. McGraw Hill, 2 edition, 1999.

[83] C. Ramchandani. Analysis of asynchronous concurrent systems by timed petri nets. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[84] A. Rozinat, I. S. M. de Jong, C. W. Gunther, and W. M. P. van der Aalst. Process mining of test processes: A case study. Technical report, WP220 Beta Paper Series Eindhoven University of Technology, 2007.

[85] N. Russell, A.H.M. ter Hofstede, D. Edmond, and WMP van der Aalst. Workflow data patterns. *QUT Technical report, FIT-TR-2004-01, Queensland University of Technology*, 2004.

[86] N. Russell, A.H.M. ter Hofstede, D. Edmond, and WMP van der Aalst. Workflow resource patterns. *Working Paper Series, WP 127, Eindhoven University of Technology*, 2004.

[87] N. Russell, A.H.M. ter Hofstede, WMP van der Aalst, and N. Mulvar. Workflow control-flow patterns: A revised view. *BPM Center Report*, 2006.

[88] N. Russell, WMP van der Aalst, and A.H.M. ter Hofstede. Exception handling patterns in process-aware information systems. *BPM Center Report*, 2006.

[89] David L. Sackett, William M. C. Rosenberg, Muir J. A. Gray, Brian R. Haynes, and Scott W. Richardson. Evidence based medicine: what it is and what it isn't. *BMJ*, 312(7023):71–72, 1996.

[90] Robert J. Schalkoff. *Pattern recognition: statistical, structural and neural approaches.* John Wiley & Sons, Inc., New York, NY, USA, 1991.

[91] Michael Sipser. *Introduction to the Theory of Computation.* PWS, 1997.

[92] Jaroslav Sklenar. Simulation of queueing networks in petrisim. In *Proceedings of the 16th European Simulation Multiconference on Modelling and Simulation 2002*, pages 403–407. SCS Europe, 2002.

[93] P.L. Spath. *Eight methodologies for clinical path development.* Editorial Cline K, 3 edition, 1996.

[94] Jan Springintveld, Frits Vaandrager, and Pedro R. D'Argenio. Testing timed automata. *Theoretical Computer Science*, 254(1–2):225–257, 2001.

[95] Thomas Stauner. Discrete-time refinement of hybrid automata. In *HSCC*, pages 407–420, 2002.

[96] P. David Stotts and William Pugh. Parallel finite automata for modeling concurrent software systems. *The Journal of Systems and Software*, 27(1):27–43, October 1994.

[97] SE Straus, WS Richardson, Paul Glasziou, and RB Haynes. *Evidence-based Medicine:How to Practice and Teach EBM.* Churchill Livingstone, 3 edition, 2005.

[98] MT System. Moviltime: http://www.mobilitime.com/sommaire.htm.

[99] Enhydra Team. Enhydra shark site: http://www.enhydra.org/index.php.

[100] WfMOpen Team. Wfmopen site: http://wfmopen.sourceforge.net/.

[101] Vicente Traver, **Carlos Fernandez**, Juan Carlos Naranjo, Eduardo Monton, Sergio Guillen, and Bernardo Valdivieso. Sistemas m-health: la solución para las necesidades de una unidad de hospitalización a domicilio. *I+S Informatica y Salud (ISSN 1579-8070)*, 44:43–49, 2004.

[102] Vicente Traver, Susana Pomés, **Carlos Fernandez**, José Conca, Lucas Sanjuán, Eduardo Roldan, Javier Berrio, Maria José Nodal, Beatriz Albella, Manuel Perez, and Sergio Guillen. Lyra: Sistema de gestión integrado para una unidad de hospitalización a domicilio. *I+S Informatica y Salud: Especial Comunicaciones en Sanidad (ISSN 1579-8070)*, 61:20–24, 2007.

[103] W. van der Aalst and A. Hofstede. Yawl: Yet another workflow language. *Information Systems*, 30:245–275, 2005.

[104] W. van der Aalst, A. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16:1128 – 1142, 2004.

[105] Wil M. P. van der Aalst, Alistair P. Barros, Arthur H. M. ter Hofstede, and Bartek Kiepuszewski. Workflow patterns. *Distributed and Parallel Databases*, page 70, 2003.

[106] Wil M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of issues and aproaches. *Data and Knowledge Engineering 47 2003*, pages 237–267, 2003.

[107] B. F. van Dongen and W.M.P. van der Aalst. Multi-phase process mining: Building instance graphs. *Lecture Notes in Computer Science,Conceptual Modeling Ű ER 2004*, 3288(1):362–376, 2004.

[108] Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and Wil M. P. van der Aalst. The prom framework: A new era in process mining tool support. In *ICATPN*, pages 444–454, 2005.

[109] A. J. M. M. Weijters, W. M. P. van der Aalst, and A. K. Alves de Medeiros. Process mining with the heuristics miner algorithm. Technical report, WP166 Beta Paper Series Eindhoven University of Technology, 2006.

[110] WfMC. *Workflow Management Coalition Terminology Glossary*. WFMC-TC-1011, Document Status Issue 3.0, 1999.

[111] W.C. Willett, L. Sampson, M.J. Stampfer, B. Rosner, C. Bain, J. Witschi, C.H. Hennekens, and F.E. Speizer. Reproducibility and validity of a semiquantitative food frequency questionnaire. *Am J Epidemiol*, 122:51–65, 1985.