UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Aplicación de técnicas de Process Mining para análisis de procesos de negocios desplegados en un BPMS.
Autores: Virginia María Magliano (9430/1).
Director: Mg. Patricia Bazán.
Asesor profesional: Lic. José Nicolás Martinez Garro
Carrera: Licenciatura en Sistemas – Plan 2007.

Resumen

En la actualidad las organizaciones usan tecnologías de información para soportar sus procesos de negocio siendo BPM (Business Process Management) la tecnología pionera. Por otro lado para analizar y optimizar los procesos y ayudar en la toma de decisiones se suelen utilizar herramientas de Business Intelligence (BI) que utilizan datos de eventos pero solo se focalizan en los datos y no en el proceso de inicio a fin. Para asistir el análisis del ciclo de vida de BPM, es necesario contar con una tecnología específica que se encuentre centrada en el proceso y no en los datos como la mayoría de los enfoques tradicionales. Es aquí cuando entra en juego Process Mining, una poderosa tecnología para administrar procesos operacionales no triviales. La principal motivación de este trabajo es proponer un nuevo enfoque para la utilización de Process Mining como metodología de análisis y optimización de procesos desplegados y ejecutados en un BPMS.

Palabras Claves

- *Process Mining*
- *Business Intelligence*
- *BPMS (Business Process Management Systems)*
- *Log de Eventos*
- *Conectores*
- *Análisis*
- *Optimización*

Conclusiones

La propuesta para la aplicación de Process Mining sobre un proceso ya implantado en un BPMS y la implementación del proceso generador del log de eventos (fuente de entrada para aplicación de las técnicas de Process Mining) permite utilizar más fácilmente Process Mining como tecnología de apoyo para analizar y optimizar procesos de negocio en cada etapa del ciclo de vida de BPM.

Trabajos Realizados

- Propuesta de un enfoque para la aplicación de Process Mining sobre un proceso ya implantado en un BPMS.
- Desarrollo de un mecanismo de integración entre Bonita Open Solution y ProM, sistematizando la construcción del log de eventos extendiendo la funcionalidad del BPMS.
- Implementación un proceso de negocio como caso de estudio en la herramienta Bonita Open Solution para luego utilizar el conector generado en el punto anterior y poder aplicar las técnicas de Process Mining con ProM.

Trabajos Futuros

- *Generar log de eventos de instancias de procesos no finalizadas y de esta forma utilizar Process Mining para hacer predicciones y análisis en línea.*
- *Automatizar la optimización de procesos que pueden proveer las técnicas de Process Mining sobre el BPMS.*
- *Análisis de otras herramientas de Process Mining y capacidades que brindan las mismas para integración con BPMS.*

Fecha de la presentación: Septiembre 2013

Agradecimientos

Quiero agradecer principalmente a Patricia Bazán y a José Martinez Garro por guiarme e incentivarme en la elaboración y culminación de este proyecto.

También quiero agradecer al proyecto Aknotec Labs y a todo el equipo de Aknotec por su cálido recibimiento, y la gran oportunidad que me han presentado.

Gracias a mi familia, por su apoyo incondicional que me permite a los 24 años estar recibiéndome de licenciada.

Finalmente, quiero gradecer a la Universidad Nacional de La Plata y a todo su equipo por formarme y acompañarme en esta hermosa etapa de la vida.

Índice General

Índice de imágenes

Índice de tablas

Introducción

En la actualidad las organizaciones usan tecnologías de información para soportar sus procesos de negocio siendo BPM (*Business Process Management* o Gestión de procesos de negocio) la tecnología pionera [1].

Por otro lado para analizar y optimizar los procesos y ayudar en la toma de decisiones se suelen utilizar herramientas de *Business Intelligence* (BI) que utilizan datos de eventos. Bajo la órbita de BI se encuentran otras tecnologías, como BAM (Monitoreo de actividades de negocio), CPM (Gestión del rendimiento corporativo), CPI (Mejora continua de procesos), y BPI (Inteligencia de procesos de negocio) que permiten realizar reportes y dashboards (tableros de mando), pero solo se focalizan en los datos y no en el proceso de inicio a fin [5].

Los BPMS (*Business Process Management Systems* o Sistemas de Administración de Procesos de Negocio) utilizan modelos de proceso para analizar y optimizar procesos operacionales. Por lo general estos modelos se encuentran desconectados de los datos de eventos reales, y por lo tanto los resultados pueden ser no confiables ya que los mismos se basan en un modelo idealizado y no en los hechos observados [5] [4] [16].

Para asistir el análisis del ciclo de vida de BPM, es necesario contar con una tecnología específica que se encuentre centrada en el proceso y no en los datos como la mayoría de los enfoques tradicionales. Es aquí cuando entra en juego *Process Mining*, una poderosa tecnología para administrar procesos operacionales no triviales. Los algoritmos y técnicas emergentes hacen posible analizar datos de eventos complejos y alinear los procesos con la información para adecuarse a los requerimientos de cliente como cumplimiento, eficiencia entre otros [3] [4] [5] [6] [16].

El objetivo de *Process Mining* es extraer conocimiento de logs de eventos obtenidos de distintos sistemas de información, para esto posee un conjunto de técnicas que se agrupan según su funcionalidad en:

- Técnicas de descubrimiento de proceso
- Técnicas de chequeo de concordancia del proceso con la realidad
- Técnicas de extensión y mejora del proceso [3] [4] [5]

La principal motivación de este trabajo es proponer un nuevo enfoque para la utilización de *Process Mining* como metodología de análisis y optimización de procesos desplegados y ejecutados en un BPMS.

Por otro lado las herramientas para la construcción del log de eventos son externas al BPMS y requieren de un conocimiento profundo de la ubicación de los datos en la base de datos del BPMS lo cual puede resultar complejo, ya que cada BPMS tiene su propia organización y administración de la base de datos [9] [11] [15], por este motivo surge interés en buscar una alternativa para la automatización de la extracción de logs de eventos desde dentro de un BPMS, agregándole funcionalidad al mismo.

La organización de este trabajo está dividida en 5 capítulos, en los que trataremos los siguientes temas:

- **Capítulo 1**: se verá un amplio marco conceptual de la tecnología *Process Mining* donde se detalla una comparación con *Business Intelligence*, y luego se hace hincapié en sus fundamentos y sus distintas técnicas.

- **Capítulo 2**: se verá el marco conceptual de BPMS y de las características principales de los mismos que se deben tener en cuenta a la hora de la integración de las tecnologías.
- **Capítulo 3**: se realizará el análisis y la propuesta de un enfoque sobre *Process Mining,* como medio para analizar los procesos de negocios ya implantados en un BPMS.
- **Capítulo 4**: se describe un mecanismo de integración entre Bonita Open Solution y ProM herramientas seleccionadas para automatizar la generación del log de eventos.
- **Capítulo 5**: se verá la implementación de un proceso de negocio como caso de estudio.

Capítulo 1: Marco conceptual de *Process Mining*

Hoy en día, los sistemas de información se encuentran cada vez más entrelazados con los procesos operacionales que soportan. Como resultado, millones de eventos se registran en los sistemas de información. Sin embargo, las organizaciones tienen problemas para extraer valor de esos datos. El objetivo de *Process Mining* es usar datos de eventos para extraer información relacionada con el proceso, es decir, para descubrir automáticamente modelos de procesos mediante la observación los eventos registrados en algún sistema de la empresa.

El crecimiento del universo digital que está bien alineado con los procesos en una organización hace posible almacenar y analizar eventos. El desafío está en explotar los datos de manera significativa, por ejemplo para proveer puntos de vista, identificar cuellos de botella, anticipar problemas, registrar violaciones de políticas, sobre todo lo dicho anteriormente se encarga *Process Mining*. [4]

1.1 *Process Mining* y definiciones

Process Mining es una disciplina de investigación joven que se encuentra entre el aprendizaje automático y la minería de datos por un lado y el modelado de procesos y análisis por el otro. La idea de *Process Mining* es descubrir, monitorear, y mejorar los procesos reales, mediante la extracción de conocimiento del log de eventos que ya se encuentra disponible en los sistemas de hoy.

En la Figura 1 se muestra que *Process Mining* establece conexiones entre los procesos actuales con sus datos por un lado y con los modelos de procesos por el otro. El universo digital y el universo físico se alinean cada vez más. Los sistemas de información registran cantidades enormes de eventos. Una gran cantidad de BPMS, WFM (Work Force Management), CRM [1](Customer Relationship Management), etc. proveen el log de eventos. Sin embargo la mayoría de los sistemas de información almacenan la información de manera no estructurada. En estos casos los datos de evento existen pero se requiere de algún esfuerzo para extraerlos. [5]

Los logs de eventos se pueden utilizar para conducir tres tipos de técnicas de *Process Mining*:

- *Descubrimiento* - Las técnicas de descubrimiento toman un log de eventos y producen un modelo sin la ayuda de ninguna información a priori. Un ejemplo es el algoritmo α. Este algoritmo toma como entrada un log de eventos y produce una red de Petri que explica el comportamiento registrado en el log. Si el log de eventos contiene información sobre los recursos, se puede descubrir también modelos de relación de recursos, es decir una red de trabajo social que muestra como las personas trabajan juntas en una organización.
- *Concordancia* - Estas comparan un modelo de proceso existente con un log de eventos del mismo proceso. El chequeo de concordancia se puede utilizar para chequear si la realidad, como se registra en el log, se corresponde con el modelo y

[1] CRM Administración basada en la relación con los clientes. CRM es un modelo de gestión de toda la organización, basada en la satisfacción del cliente.

viceversa. El chequeo de concordancia se debe utilizar para detectar, localizar y explicar desviaciones, y para medir la severidad de estas desviaciones.

- *Mejora* - Aquí, la idea es extender o mejorar un modelo de proceso existente utilizando información sobre el proceso actual registrado en algún log de eventos. Mientras que el chequeo de concordancia mide la alineación entre el modelo y la realidad, este tercer tipo de técnica de *Process Mining* tiene como objetivo cambiar o extender el modelo a priori. Un tipo de mejora es la reparación, es decir modificar el modelo para reflejar mejor los resultados de la realidad. Por ejemplo, si dos actividades están modeladas secuencialmente pero en la realidad pueden ejecutarse en cualquier orden, el modelo debe ser modificado para reflejar esto. Otro tipo de mejora es la extensión, es decir, agregar una nueva perspectiva al modelo de proceso por correlación cruzada con el log. Un ejemplo es la extensión de modelos de procesos con datos de performance. Por ejemplo, utilizando *timestamps* en el log de eventos, para mostrar cuellos de botella, servicios, niveles y frecuencias.

Ortogonalmente a los tres tipos de *Process Mining* (descubrimiento, concordancia y mejora) se pueden identificar distintas perspectivas. De ahora en adelante consideraremos las siguientes perspectivas:

- *La perspectiva de control de flujo*, se focaliza precisamente en el control de flujo, es decir en el orden de las actividades. El objetivo de obtener esta perspectiva es encontrar una buena caracterización de los posibles caminos. Expresado en términos de una red de Petri o alguna otra notación como BPMN.
- La *perspectiva organizacional* se focaliza en información sobre los recursos que se encuentran en el log, es decir que actores se involucran y como se relacionan. El objetivo es estructurar la organización mediante la clasificación de las personas en términos de roles y unidades organizacionales, o mostrar la red de trabajo social.
- *La perspectiva de caso* se focaliza en las propiedades de caso. Los casos se pueden caracterizar por los valores de los elementos de datos correspondientes. Como por ejemplo el número de productos encargados.
- La *perspectiva de tiempo* es considerada con el tiempo y la frecuencia de los eventos. Cuando los eventos contienen timestamps es posible descubrir cuellos de botella, medir niveles de servicio, monitorear la utilización de los recursos y predecir el tiempo restante de procesamiento de los casos en ejecución.

Las diferentes perspectivas se encuentran sobrepuestas y proveen una buena caracterización de los aspectos que *Process Mining* tiene como objetivo analizar.

Process Mining realiza análisis fuera de línea, es decir los procesos se analizan luego de su ejecución para analizar cómo se pueden mejorar o tener un mejor entendimiento. Pero también las técnicas de *Process Mining* también se pueden utilizar en línea. Nos referimos a esto como soporte operacional. Un ejemplo puede ser la detección de no concordancia en el momento que la desviación está siendo llevada a cabo. Otro ejemplo es para predicción de tiempos de los casos que se están ejecutando, es decir dado un caso ejecutado parcialmente el tiempo restante de procesamiento se puede estimar basándose en la información histórica de casos similares.

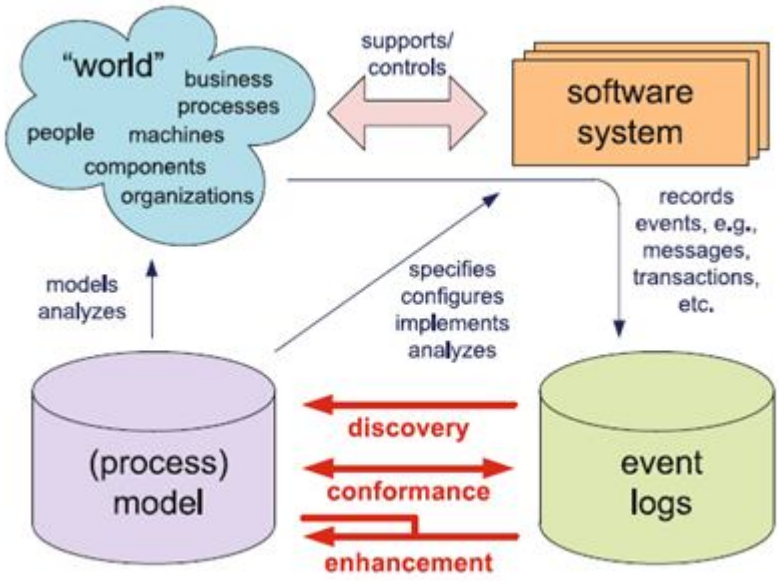


Figura 1 Ubicación de los tres tipos de *Process Mining*: Discovery, concordancia y mejora. [5]

1.1.1 Play-in, Play-out, and Replay

Uno de los elementos principales de *Process Mining* es el énfasis en establecer una relación fuerte entre el modelo de proceso y la "realidad" capturada en la forma de un log de eventos. Utilizamos los términos Play-in, Play-out, y Replay para reflejar esta relación. La Figura 2 expresa estas tres relaciones:

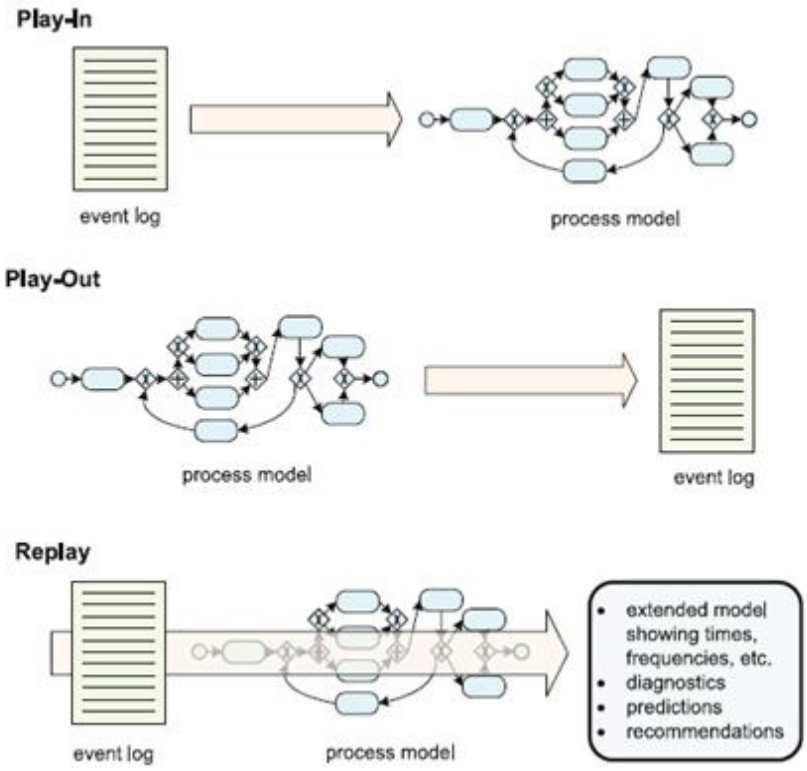


Figura 2 Tres formas de relacionar log de eventos (u otras fuentes de información que contengan comportamiento de ejemplo) con modelos de procesos: Playin, PlayOut, replay. [5]

- *Play-in* dado un log de eventos se puede deducir un comportamiento y generar un modelo. Play-in también se refiere como deducción. La mayoría de las técnicas de *data mining* utilizan Play-in un modelo se diseña en base a los ejemplos (al log de eventos).
- *Play-out* se refiere al uso clásico de modelos de procesos. Dado un modelo se puede generar comportamiento que se almacena en un log de eventos. *Pay-out* se puede usar para aprobar procesos operacionales utilizando algún modelo ejecutable. Las herramientas de simulación también utilizan un motor Play-out para conducir experimentos. La idea principal de la simulación es correr repetidamente un modelo y recorrer las estadísticas y los intervalos confidentes. Un motor de simulación es similar a un motor de *workflow*. La diferencia principal es que el motor de simulación interactúa con un ambiente modelado mientras que el motor de *workflow* interactúa con un ambiente real.
- *Replay* utiliza tanto un log de eventos como un modelo como entrada. Un log de eventos puede volverse a correr por diferentes propósitos:
    - Chequeo de concordancia: las diferencias entre el modelo y el log se pueden detectar volviendo a correr el log sobre el modelo.
    - Extender el modelo con frecuencias e información temporal: Cuando se vuelve a correr el log se puede ver que partes del modelo se visitan con frecuencia. También se puede utilizar para detectar cuellos de botella.

- Construir modelos predictivos: para diferentes estados del modelo se pueden realizar predicciones particulares.
- Soporte operacional: Replay no está limitado a datos de eventos históricos. Se pueden volver a correr trozos parciales de casos que se encuentran corriendo. Esto se puede usar para detectar desviaciones en momento de ejecución.[3][4]

1.1.2 *Business Intelligence* y *Process Mining*

Si bien, *Process Mining* es una herramienta poderosa dentro del amplio contexto de BPM, se suele considerar bajo el campo de *Business Intelligence*, sin embargo, existen diferencias entre estas dos tecnologías que veremos en el siguiente apartado.

BI es un concepto amplio que incluye todo lo que tenga como objetivo proveer información accionable que se utilice para la toma de decisiones. *Process Mining* se puede ver como una colección de técnicas de BI. Sin embargo, la mayoría de las herramientas de BI no son realmente inteligentes y no proveen ninguna de las capacidades de *Process Mining*, se focalizan en consultar y generar reportes, combinados con técnicas de visualización que muestran dashboards (gráficos).

Forrester define a *Business Intelligence* en dos maneras. La definición principal dice "BI es un conjunto de metodologías. procesos, arquitecturas y tecnologías que transforman los datos en bruto en información útil y con significado que se usa para habilitar puntos de vista operacionales y toma de decisiones más efectivas, estratégicas y tácticas". La segunda definición provista por Forrester es más estrecha: "BI es un conjunto de metodologías, procesos, arquitecturas y tecnologías que aprovechan la salida de procesos de administración de información para analizar, reportar, administración de performance y delivery de información".
El mercado de productos BI está creciendo y madurando. Los productos de BI más ampliamente usados son los siguientes:

Herramienta	**Propietario**	**Link**
IBM Cognos *Business Intelligence*	IBM	http://www-01.ibm.com/software/analytics/cognos/
Oracle *Business Intelligence*	Oracle	http://www.oracle.com/us/solutions/business-analytics/business-intelligence/overview/index.html
SAP BusinessObjects	SAP	http://www.sap.com/spain/pc/analytics/business-intelligence.html
WebFOCUS	Information Builders	http://www.informationbuilders.es/products/webfocus
MS SQL Server	Microsoft	http://www.microsoft.com/en-us/server-cloud/products/sql-server-editions/sql-server-business-intelligence.aspx
MicroStrategy	MicroStrategy	http://www.microstrategy.com.ar/

NovaView	Panorama Software	http://www.panorama.com/blog/press-releases/panorama-software-releases-novaview-5-5-the-first-bi-suite-to-combine-on-premise-with-saas-bi-for-the-enterprise/
QlikView	QlikTech	http://www.qlik.com/
SAS Enterprise *Business Intelligence*	SAS	http://www.sas.com/en_us/software/business-intelligence/enterprise-bi-server.html
TIBCO Spotfire Analytics	TIBCO	http://spotfire.tibco.com/
Jaspersoft	Jaspersoft	https://www.jaspersoft.com/es
Pentaho BI Suite	Pentaho	http://www.pentaho.com/

Las funcionalidades típicas provistas por estos productos incluyen:

- ETL (*Extract, Transform, and Load*). Todos los productos soportan la extracción de datos de varias fuentes. Los datos extraídos se transforman a un formato de datos estándar (por lo general una tabla multidimensional) y cargados en un sistema de BI.
- Consultas Ad-hoc. Los usuarios pueden explorar los datos de manera personalizada.
- Reportes. todos los productos de BI proveen la definición de reportes estándares.
- *Dashboards* interactivos. Todos los productos de BI permiten la definición de dashboards que consisten de datos tabulares y una variedad de gráficos. Estos dashboards son interactivos esto es que el usuario puede cambiar, refinar, agregar y filtrar la vista actual utilizando controles predefinidos.
- Generación de alertas. Es posible definir eventos y condiciones que puedan disparar una alerta por ejemplo, cuando las compras caen a un determinado índice un e-mail se manda al gerente de ventas.

La siguiente tabla comparativa muestra las diferencias principales entre *Business Intelligence* y *Process Mining:*

Tabla 1 Comparación *Business Intelligence* con *Process Mining*

	Herramienta BI	**Herramienta *Process Mining***
Entrada	Datos relacionales, el orden de los eventos se pierde	Log de eventos, se puede analizar el proceso de inicio a fin
Reportes	Amplio rango de gráficos	Técnicas de análisis de bajo nivel, que contemplan la totalidad del proceso
Objetivo Principal	Monitoreo, Dashboards y reportes	Análisis profundo del proceso
Alcance	Limitado a datos históricos	Permite análisis en línea, predicciones y recomendaciones para un proceso en ejecución

Los enfoques administrativos, como *Continuous Process Improvement* (CPI), Total *Quality Management* (TQM) y *Six Sigma* también están relacionados con *Process Mining*. Estos enfoques tienen en común que los procesos son minuciosamente analizados para identificar si se pueden incorporar posibles mejoras. Claramente *Process Mining* contribuye en el análisis de desviaciones e ineficiencias. [4]

Ahora que hemos visto una breve descripción de lo que es esta tecnología entraremos en detalle en sus fundamentos y los distintos tipos de técnicas que *Process Mining* nos provee para analizar procesos (Descubrimiento de procesos, chequeo de concordancia y extensión del modelo).

1.2 Fundamentos de *Process Mining*

Para comprender el objetivo de esta disciplina haremos una breve descripción de sus fundamentos. En esencia *Process Mining* se compone por dos pilares:

- Modelado y análisis de procesos.
- Data mining

En las próximas secciones describiremos el análisis y el modelado de procesos y las técnicas de data mining sobre las que se tiene sus orígenes *Process Mining*.

1.2.1 Modelado y análisis de proceso

La cantidad de notaciones de modelado de procesos existentes hoy en día nos demuestra la relevancia de los modelos de proceso. Algunas organizaciones pueden usar solo modelos de procesos informales para estructurar las discusiones y para documentar los procedimientos. Sin embargo, las organizaciones que operan en un nivel más alto de BPM usan modelos que se pueden analizar y usar para aprobar modelos operacionales. [4]

Hoy en día, las innovaciones en computación y en comunicación son los mayores productores de cambios en los procesos de negocio. Los procesos de negocio se han vuelto más complejos. Los modelos de procesos asisten en la administración de la complejidad proveyendo puntos de vista y documentando los procedimientos. Los sistemas de información se deben configurar y manejar por instrucciones precisas. Los procesos que

atraviesan la organización pueden funcionar adecuadamente si se hace un acuerdo común en las interacciones requeridas. Como resultado, los modelos de procesos se utilizan ampliamente en las organizaciones.

Los modelos se usan para razonar sobre los procesos (rediseño) y para tomar decisiones dentro del proceso (planificar y controlar). Un modelo de proceso expresado en BPMN se puede utilizar para discutir responsabilidades, analizar el cumplimiento, predecir la performance utilizando simulación y configurar un sistema WFM.

Realizar un buen modelo en BPM es más un arte que una ciencia. Durante el modelado de un proceso se puede producir distintos tipos de errores:

- El modelo puede describir una versión idealizada de la realidad.
- Incapacidad de capturar adecuadamente el comportamiento humano. No se puede modelar el proceso aislado sino que se tiene que tener en cuenta las distintas intervenciones que tiene el usuario en distintas tareas del proceso, o en otros procesos.
- El modelo se puede encontrar en un nivel de abstracción equivocado. Dependiendo de los datos de entrada y las preguntas que se requieren responder se debe elegir un nivel de abstracción adecuado. Puede ser que el modelo sea muy abstracto y q sea incapaz de responder preguntas muy concretas, o puede ser muy detallado y así costar la comprensión

Estos son algunos de los problemas que una organización enfrenta cuando se hacen modelos a mano. Solo diseñadores y analistas con experiencia pueden hacer modelos que tengan un valor predictivo bueno y se pueden usar como punto de entrada para (re) implementación o rediseño. Un modelo inadecuado puede llevar a conclusiones incorrectas. Por lo tanto se avoca al uso de datos de eventos. *Process Mining* permite la extracción de modelos basados en hechos. Además, *Process Mining* no tiene como objetivo crear un solo modelo del proceso. En cambio provee varias vistas de la misma realidad en diferentes niveles de abstracción. Por ejemplo se puede inspeccionar el comportamiento más frecuente. *Process Mining* puede también revelar que las personas en una organización no funcionan como maquinas. Por un lado puede mostrar que se producen muchos tipos de ineficiencias, y por otro puede mostrar la flexibilidad remarcable de algunos empleados para enfrentarse con problemas y distintos trabajos.

Process Mining permite la extracción de modelos basado en hechos (log de eventos). [4]

Modelos de procesos

Modelar procesos no es una tarea fácil, pero sin embargo es una tarea muy importante y necesaria. *Process Mining* puede facilitar la construcción de modelos mejores en menos tiempo. Los algoritmos de descubrimiento de procesos como el algoritmo α pueden generar automáticamente un modelo de proceso. Si bien ciertas técnicas de *Process Mining* dan como resultado un modelo de proceso especifico, se pueden convertir fácilmente un modelo en de una notación a otra. Por ejemplo el algoritmo α produce una red de Petri como resultado y se puede convertir fácilmente en un modelo BPMN, BPEL, o un diagrama de actividades UML.

Process Mining utiliza modelos de procesos en distintas notaciones, las cuales son: sistemas de transiciones, redes de Petri, redes de trabajo social. YAWL[2] (Yet Another Workflow Language), BPMN[3](Business Process Management Notation), redes casuals entre otros.

Para el modelado de procesos nos concentraremos en la perspectiva de control de flujo (mencionada en la sección *Process Mining* y definiciones). Asumimos que existe un conjunto de nombres de actividades *A*. El objetivo de un modelo de proceso es decidir cuales actividades deben ejecutarse y en qué orden. Las actividades pueden ejecutarse secuencialmente, algunas pueden ser opcionales, otras concurrentes y es posible la repetición de la misma actividad. [4]

Análisis de proceso basado en modelo

Los principales enfoques para análisis basado en modelos son: verificación y análisis de performance. La verificación se interesa en la correctitud de un sistema o proceso. El análisis de performance se focaliza en los tiempos de flujos, los tiempos de espera, la utilización y niveles de servicios.

Verificación: Cuando una red de trabajo tiene caminos permitidos erróneos, se dice que el modelo tiene ruido, una tarea de verificación puede ser detectar el ruido en el modelo. Otra tarea de verificación puede ser la comparación entre dos modelos, por ejemplo se puede comparar la implementación de las necesidades de un proceso con una especificación de más alto nivel del mismo proceso.

Análisis de performance: por lo general se identifican 3 dimensiones de performance: tiempo, costo y calidad. Para cada uno de estas dimensiones de performance se pueden definir distintos Key Performance Indicitos (KPI).

Cuando miramos a la dimensión tiempo se pueden definir los siguientes indicadores de performance:

- El tiempo de flujo: es el tiempo total desde la creación del caso hasta la finalización del mismo. Se pueden medir el promedio del tiempo de flujo de todos los casos. Sin embargo el nivel de varianza puede ser importante y debe medirse también.
- El tiempo de servicio es el tiempo que realmente requiere un caso. Por lo genera el tiempo de servicio es una fracción del tiempo de flujo (o tiempo legado).
- El tiempo de espera, es el tiempo que un caso espera para que un recurso esté disponible. Este tiempo se puede calcular para cada actividad o para un caso entero.

---

[2] YAWL (Yet Another Workflow Language) es un lenguaje de workflow basado en los patrones de Workflow. Este lenguaje está soportado por un sistema de software que incluye un motor de ejecución y un editor gráfico.

[3] BPMN (Standard Business Process Model and Notation) es una notación grafica estandarizada que permite el modelado de procesos de negocio.

- El tiempo de sincronización es el tiempo que una actividad no está completamente habilitada y espera un disparador externo o algún acontecimiento paralelo. El caso espera por sincronización en vez de por un recurso.

Los indicadores de performance se definen también para la dimensión de costo. El costo de ejecutar una actividad puede depender del tipo de recurso utilizado, la utilización, o la duración de una actividad. El costo de los recursos puede depender de la utilización de los mismos. Un índice indicador de performance en la mayoría de los procesos es el la utilización promedio de los recursos sobre un período de tiempo.

La dimensión de calidad por lo general se focaliza en el producto o el servicio que se provee al cliente. Como los costos se puede medir de diferentes formas. Se puede medir por ejemplo el número promedio de quejas por caso o el número de defectos por producto.

Mientras que la verificación se focaliza en la correctitud lógica del proceso modelado, el objetivo del análisis de performance es mejorar los procesos con respecto al tiempo, costo y calidad.

Por lo general los modelos analíticos requieren muchas suposiciones y pueden solo utilizarse para cuestiones particulares. Por lo tanto a veces se debe acudir a la simulación. La mayoría de las herramientas de BPM proveen simulación.

Muy pocas herramientas utilizan la simulación de manera efectiva, esto puede suceder por falta de entrenamiento y limitaciones de las herramientas existentes. Pero también hay problemas fundamentales en la simulación, los modelos de simulación tienden a sobre simplificar las cosas, por ejemplo las personas no trabajan a velocidades constantes, y muchas veces requieren de distribuir su atención entre muchos procesos. Esto puede traer efectos dramáticos a la performance de un proceso y por eso esos aspectos se deben tener en cuenta.

Los modelos deben coincidir con la realidad, porque de no ser así se estarían analizando y realizando simulaciones con modelos que son "idealizados" y hay poca alineación entre los modelos realizados y la realidad. El objetivo de *Process Mining* es solucionar estos problemas estableciendo una conexión directa entre los modelos y los datos de eventos de bajo nivel actuales del proceso. Las distintas técnicas de descubrimiento, permiten ver la misma realidad en diferentes ángulos y en diferentes niveles de abstracción. [4]

1.2.2 Data mining

Las ideas originarias del campo de data mining son utilizadas para la evaluación de resultados de *Process Mining*. Por ejemplo uno puede adoptar varias aproximaciones de data mining para medir la calidad de modelos de procesos mejorados o descubiertos. Un entendimiento básico de data mining puede ayudar al entendimiento completo de las técnicas de *Process Mining*.

Data mining se define como "el análisis de (por lo general grandes) conjuntos de datos para encontrar relaciones no sospechadas y resumir los datos a maneras nobles que sean comprensibles y útiles para los dueños de los datos". La entrada es típicamente una tabla y la salida puede ser las reglas, los clusters, estructuras de árboles, gráficos, ecuaciones, patrones, etc. El crecimiento del universo digital es el principal conductor de la popularidad de data mining.

Las tablas están compuestas por filas que representan instancias, individuos, entidades,

casos objetos o registros. Las columnas son llamadas variables. Las variables son referidas por lo general como atributos, características, o elementos de datos.
Se hace una distinción entre *variables categóricas* y *variables numéricas*. Las variables categóricas tienen un conjunto limitado de valores posibles, y pueden enumerarse fácilmente por ejemplo una variable booleana que puede ser true o false. Las variables numéricas tienen un orden y no se pueden enumerar fácilmente. Por ejemplo temperatura, edad, peso, número de tiempos y altitud.
Las variables categóricas se dividen en *ordinales* y *nominales*. Las variables nominales no tienen orden lógico. Por ejemplo los booleanos los colores, países, no tienen un acuerdo común para un orden lógico. Las variables ordinales tienen un orden asociado. Por ejemplo la calificación (excelente, bueno, malo).
Por lo general, antes de aplicar cualquier técnica de data mining los datos son pre procesados. Es decir las columnas y las filas pueden ser removidas por varias razones. Las instancias que están corruptas deben ser removidas también. Además el valor de una variable para una instancia particular puede estar perdido o tener el valor equivocado. Las técnicas de data mining no realizan tantos supuestos sobre el formato de la entrada como las técnicas de *Process Mining*. En *Process Mining* los eventos se ordenan de acuerdo al tiempo, mientras que en data mining las filas no tienen ningún orden. Para casos particulares es posible convertir un log de eventos en un conjunto de datos simples para data mining. Nos referiremos a esto como extracción de características. Luego utilizaremos la extracción de características para varios propósitos como ser análisis de decisiones, en un modelo de proceso descubierto y clusterización de casos antes del descubrimiento de proceso para que de esa forma cada clúster tenga un modelo de proceso dedicado.
Luego de estudiar la estructura básica de entrada para data mining, clasificamos las técnicas de data mining en dos categorías principales: aprendizaje supervisado, y aprendizaje no supervisado. [3][4]

Aprendizaje supervisado: clasificación y regresión

El aprendizaje supervisado asume datos etiquetados, es decir, hay una variable respuesta que etiqueta cada instancia. Las otras variables son variables predictivas y estamos interesados en explicar las variables respuesta en términos de las variables predictivas. A veces las variables de respuesta son llamadas variables dependientes y las variables predictivas se llaman variables independientes. Por ejemplo, queremos predecir el resultado final de un estudiante en términos de las notas de los estudiantes del curso.
Las técnicas *de aprendizaje supervisado* se pueden dividir en *clasificación* y *regresión* dependiendo del tipo de variable respuesta (categórica o numérica)
Las técnicas de clasificación asumen una respuesta categórica y el objetivo es clasificar instancias basadas en las variables predictivas.
Las técnicas de regresión asumen como respuesta una variable numérica. El objetivo es encontrar una función que encaje los datos con menos errores.
La técnica de regresión más frecuente es la regresión linear. Dada una variable de respuesta *y* y las variables predictivas *x1, x2, ...xn* un modelo lineal $y = f(x1, x2, \ldots, xn) = a0 + \sum_{i=1}^{n} aixi$ se aprende sobre el conjunto de datos. Por cada instancia en el conjunto de datos hay un error de $|y - \hat{y}|$.
Un enfoque es minimizar la suma de los errores al cuadrado, es decir dadas m instancias el

objetivo es encontrar una función *f* para *que mj=1(yj − ŷj)2* sea mínimo. La clasificación requiere una variable de respuesta categórica. En algunos casos tiene sentido transformar una variable de respuesta numérica en una categórica. Por ejemplo transformar la variable numérica edad en una variable categórica que las divida en infante, joven, adulto, etc. [4]

Aprendizaje no supervisado: Clustering y descubrimiento de patrones

El aprendizaje no supervisado asume datos no etiquetados, es decir, las variables no se separan en variables de respuesta o de predicción. Analizaremos dos tipos de aprendizaje no supervisado: *clustering* y *descubrimiento de patrones.*
Los algoritmos de clustering examinan los datos para encontrar grupos de instancias que son similares. A diferencia de la clasificación, no se focaliza en algunas variables de respuesta, sino que en las instancias como un todo. Por ejemplo el objetivo puede ser encontrar grupos homogéneos de estudiantes, o clientes. Algunas técnicas bien conocidas de clustering son *k-means clustering* y *clustering jerárquico aglomerativo*.
Hay muchas técnicas para descubrir patrones en los datos. Por lo general el objetivo es encontrar reglas de la forma: *IF X THEN Y* donde *X* e *Y* relacionan valores de variables diferentes. La técnica más conocida para estas cuestiones es minería de reglas de asociación.
Los arboles de decisión también se pueden convertir en reglas. Sin embargo, un árbol de decisión se construye para una variable de respuesta particular. Por lo tanto las reglas que se extraen de un árbol de decisión solo dicen algo sobre las variables de respuesta en términos de algunas de las variables predictivas. Las reglas de asociación se descubren utilizando aprendizaje no supervisado, esto es no se requiere seleccionar una variable de respuesta.
Los resultados de data mining pueden ser o descriptivos o predictivos. Los árboles de decisiones, reglas de asociación, y funciones de regresión dicen algo del conjunto de datos usado para aprender el modelo. Sin embargo también se pueden utilizar para hacer predicciones para nuevas instancias es decir, predecir la performance total de los estudiantes basándose en las calificaciones del curso obtenidas en el primer semestre. [4]

En el anexo 1 se describirán las técnicas de data mining de aprendizaje supervisado y aprendizaje no supervisado que más influyen en las técnicas de *Process Mining*.

Las técnicas de data mining enunciadas en el anexo 1 pueden ser explotadas para *Process Mining*, pero no se pueden usar directamente para las tareas más importantes de la misma como descubrimiento de procesos, chequeo de concordancia y mejora de procesos. Sin embargo hay otra razón para mostrar esta variedad de técnicas de data mining, al igual que en data mining, para *Process Mining* no es trivial analizar la calidad de los resultados. *Process Mining* se puede beneficiar de las experiencias en el campo de data mining.

1.3 Logs de eventos, Fuentes de datos

Como se mencionó previamente el objetivo de *Process Mining* es descubrir, monitorear, y mejorar los procesos reales, mediante la extracción de conocimiento del log de eventos que ya se encuentra disponible en los sistemas de hoy.
La estructura del log de eventos puede variar de acuerdo a la técnica de data mining que se utilizará. El desafío es extraer estos datos de diferentes variedades de fuentes de datos, como ser bases de datos, sistemas de archivos planos, logs de mensajes y sistemas de

administración documentados. Cuando se mezcla y se extraen los datos la sintaxis y la semántica juegan un rol fundamental. Dependiendo de las preguntas que se buscan contestar se requieren distintas vistas de los datos disponibles.

La idea de *Process Mining* es analizar datos de eventos desde una perspectiva orientada a procesos. El objetivo es responder a preguntas de procesos operacionales. Como por ejemplo ¿Cómo controlar mejor el proceso?
En el proceso general de *Process Mining* mostrado en la Figura 3 se realiza un especial énfasis en lo rol de los datos de eventos. El punto de comienzo son los datos crudos escondidos en todo tipo de fuentes de datos. Una fuente de dato puede ser un archivo plano, una planilla de Excel, un log transaccional, o una tabla de una base de datos. Obviamente todos los datos se encuentran en diferentes formatos y formas de estructuración. Se requiere un poco de esfuerzo recolectar de todas estas fuentes la información relevante en un formato adecuado.

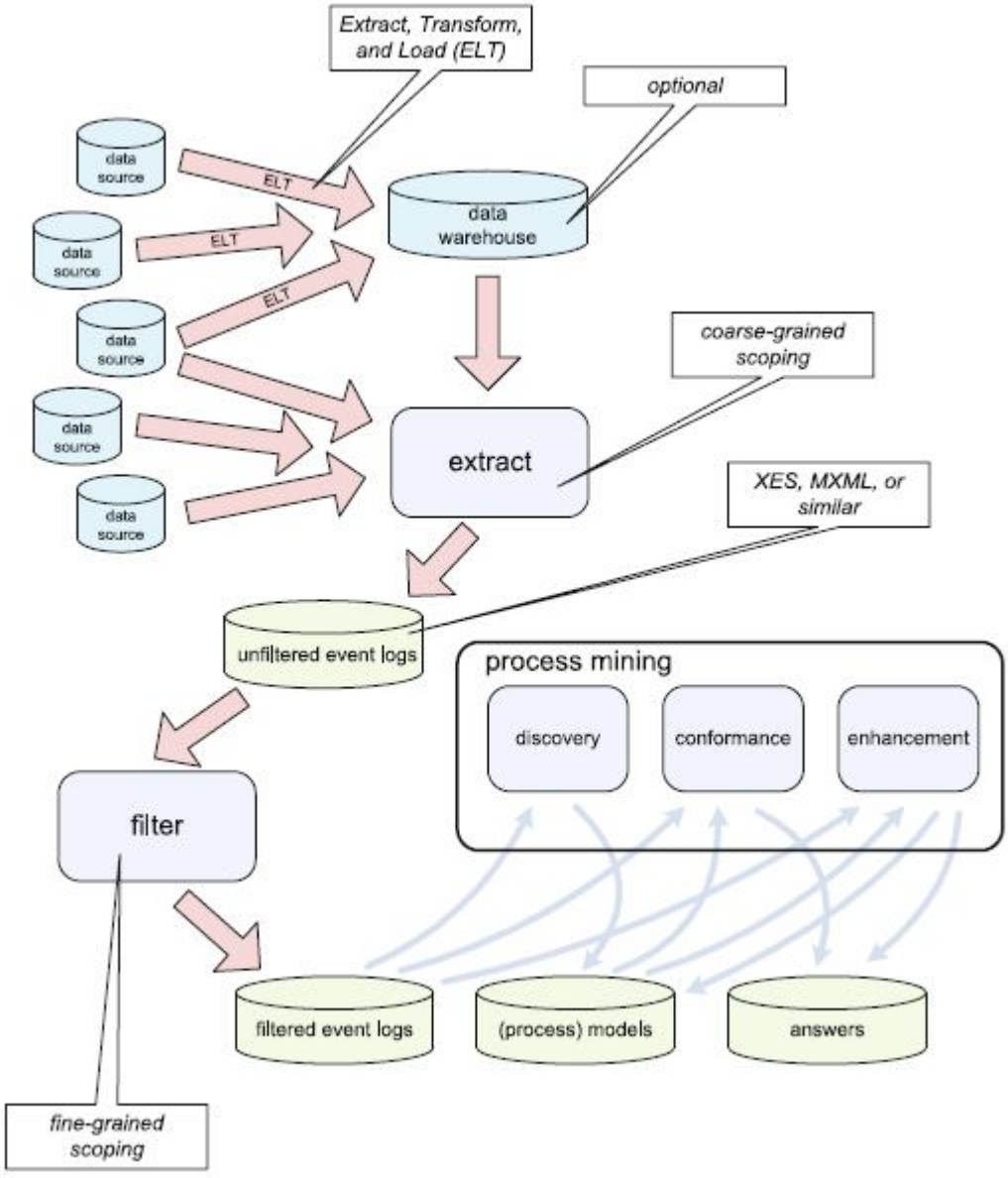


Figura 3 Visión global describiendo el flujo de trabajo desde fuentes de datos heterogéneas hasta los resultados de *Process Mining* [4]

Tanto en el contexto de BI como en el de data mining existe la fase de ETL "extract, Transform, and Load" que se utiliza para describir el proceso que involucra:
a) *extracción* de datos de una fuente externa
b) *transformación* de los datos para adecuarse a necesidades operacionales (tratando con temas de sintáctica y semántica mientras se asegura los niveles de calidad predefinidos)
c) *carga* en un sistema como un data warehouse o una base de datos relacional.
Un *data warehouse* es un repositorio lógico único de los datos transaccionales y operacionales de la organización. El data warehouse no produce datos simplemente los almacena de sistemas operacionales. El objetivo es unificar información como esa para que pueda ser usada para reporte, análisis, etc. Muchas organizaciones proveen su data warehouse pero el mismo no es orientado a procesos, por lo general el data warehouse se utiliza para Online Analytical Processing (OLAP) que no poseen mucha información relacionada al proceso, y *Process Mining* requiere de eventos relevantes de procesos y el orden de los mismos.
Haya o no un data warehouse se deben extraer los datos y deben convertirse en un log de eventos. Aquí el alcance es el más importante. Por lo general el problema no es de conversiones sintácticas sino la selección de los datos adecuados.
Los formatos típicos para almacenar el log de eventos son XES (eXtensible Event Stream) and MXML (Mining eXtensible Markup Language).
Un log de eventos corresponde a un solo proceso.
Dependiendo de las preguntas y los puntos de vista elegidos, se pueden extraer distintos logs de eventos del mismo conjunto de datos. Una vez que se crea el log de eventos por lo general se filtra. El filtrado es un proceso iterativo. El *alcance de grano grueso* se realiza cuando se extraen los datos en un log de eventos. El filtrado corresponde al *alcance de grano fino* que se basa en los resultados de análisis inicial.
Basándose en el log filtrado se pueden aplicar los distintos tipos de técnicas de *Process Mining*: descubrimiento, concordancia, y mejora. [3] [4].

Hasta hace poco el estándar por defecto para almacenar e intercambiar logs de eventos era el MXML (mining eXtensible Markup language). MXML emergió en el 2003 y luego se adoptó por ProM. Utilizando MXML se puede almacenar un log de eventos utilizando una sintaxis basada en XML.
ProMimport es una herramienta que soporta la conversión de diferentes fuentes de datos a MXML, como por ejemplo MS Access, Aris PPM, CSV, Apache, PeopleSoft, Subversion, SAP R/3. Protos, Cognos. MXML tienen una notación estándar para almacenar timestamps, recursos y tipos de transacciones. Además se pueden agregar arbitrariamente atributos a eventos y casos. El resultado posterior es un MXML ad-hoc con ciertos atributos de datos que se interpretan de una forma específica.
XES es el sucesor de MXML. Se basa en muchas experiencias prácticas que se tuvieron con MXML, el formato XES se hizo menos restrictivo y realmente extensible. Es el formato adoptado por la IEEE *Task Force* on *Process Mining*. El formato es adoptado por herramientas como ProM, Nitro, XESme, y OpenXES. [4]

En el anexo 2 se provee información sobre la estructura que debe tener el log de eventos y una descripción detallada del estándar XES.

1.4 Descubrimiento de procesos

Una vez que se obtiene el log de eventos desde los distintos sistemas de información nos encontramos en condiciones de aplicar las distintas técnicas de *Process Mining*: descubrimiento de procesos, chequeo de concordancia y extensión.

Las técnicas de descubrimiento de procesos son técnicas de tipo *Play-in* que se encargan de la perspectiva de control de flujo de proceso, es decir toma como entrada un log de eventos, y construye un modelo de proceso, capturando el comportamiento que se ve en el log.

A continuación la definición del algoritmo de descubrimiento de procesos:

Definición (problema general de descubrimiento de procesos): Sea *L* el log de eventos, especificado por el estándar XES. Un algoritmo de descubrimiento de proceso es una función que mapea *L* en un modelo de proceso, por esto es que el modelo es representativo del comportamiento que se ve en el log de eventos. El desafío es encontrar tal algoritmo. No se especifica qué tipo de modelo se generará si BPMN, YAWL o un modelo de red de Petri.

La definición anterior no especifica el formato de destino y se utiliza un log de eventos como entrada, sin especificar los requerimientos tangibles. Para hacer los requerimientos más concretos definimos el destino como una red de Petri. Además utilizaremos un log de eventos simple *L*, que es una conjunto múltiple de traces sobre un conjunto de actividades *A*, es decir, $L \in B(A*)$. Por ejemplo:

L1 =[(a, b, c, d)3, (a, c, b, d)2, (a, e, d)]

El objetivo es descubrir una red de Petri que pueda reproducir el log de eventos. Definición (problema de descubrimiento de procesos específico) un algoritmo de descubrimiento de procesos es una función γ que mapea un log $L \in B(A*)$ en una red de Petri $\gamma\ (L) = (N,M)$. Idealmente, *N* es una red de workflow y todos los traces de *L* corresponden a una posible secuencia de disparo de *(N, M)*.

La función γ define una técnica Play-in. Basándonos en el *L1*, un algoritmo de descubrimiento de proceso puede descubrir la red de workflow de la figura 4.

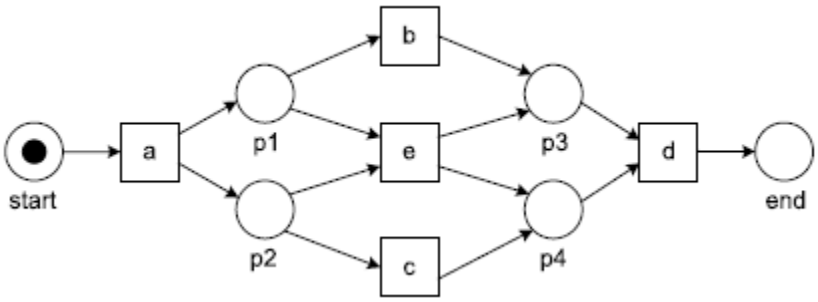


Figura 4 Red-WF N1 descubierta del L1 =[(a, b, c, d)3, (a, c, b, d)2, (a, e, d)]

Se puede ver como la red de workflow puede reproducir todas las instancias del log L1.

La mayoría de las notaciones de modelado asumen el criterio de correctitud y para todos los fenómenos como livelocks y deadlocks no son deseados, independientemente de la notación.

El formalismo compacto de la semántica formal de las redes de Petri puede ser el más

indicado para explicar los algoritmos de descubrimiento de procesos. Pero hay muchas técnicas para trasformar redes de Petris en otras notaciones como ser BPMN, YAWL, etc. El modelo descubierto debe ser representativo del comportamiento que se observa en el log de eventos. Esto se realiza bajo el requerimiento de que el modelo sea capaz de reproducir todo el comportamiento en el log, es decir cada rastro en el log de eventos es un posible disparo de secuencia del el WF-net (work-Flow net). Esto es lo que se llama requerimiento de adecuación. En general, hay un equilibrio entre los siguientes cuatro criterios de calidad:

- *Fitness*: el modelo descubierto debería permitir el comportamiento observado en el log de eventos.
- *Precisión*: El modelo descubierto no debería permitir comportamiento que no esté relacionado con lo que se vea en el log de eventos.
- *Generalización*: el modelo descubierto debe generalizar el comportamiento ejemplo que se observa en el log.
- *Simplicidad*: El modelo descubierto debe ser simple.

Un modelo que tenga buen fitness es capaz de reproducir la mayoría de los rastros en un log. La precisión está relacionada con la noción de *underfitting* presentada en el contexto de data Mining. Un modelo con precisión pobre tiene *underfitting*, es decir, permite un comportamiento que es muy diferente al que se observa en el log de eventos. La generalización está relacionada con la noción de overfitting. Un modelo con *overfitting* no generaliza lo suficiente, es decir, es muy específico y se deja llevar por los ejemplos en el log de eventos. El cuarto criterio está relacionado con la navaja de Occams que citaba que " no se debe incrementar, más de lo necesario, el número de entidades requeridas para explicar algo". Siguiendo este principio buscamos al modelo de proceso más simple.

Es un desafío balancear los cuatro criterios. Por ejemplo, un modelo muy simplificado tiende a tener menos fitness o falta de precisión. Además, hay una compensación obvia entre underfitting y overfitting. [4].

A continuación describiremos un algoritmo simple de descubrimiento de procesos sobre el que se basan las técnicas más robustas de descubrimiento de procesos. Luego en el anexo 3 se encontrara más detalle sobre este algoritmo y las técnicas de descubrimiento de procesos más conocidas.

Algoritmo α: dado un log de eventos simple produce una red de Petri que (por lo general) puede reproducir el log. Es el primer algoritmo de descubrimiento de procesos que puede representar la concurrencia. Sin embargo no es una técnica de minería muy práctica porque tiene problemas de ruido, comportamiento no frecuente o incompleto, y estructuras de ruteo complejas. El algoritmo α es simple y muchas de sus ideas fueron introducidas en técnicas más robustas y complejas como *Heurisitc Mining*, *Genetic Process Mining* y *Region Based Mining*.

La entrada de un algoritmo α es un log de eventos simple L, es decir $L \in B(A^*)$

Nos referiremos a conjuntos de actividades como A. Estas actividades corresponderán a transiciones en la red de Petri descubierta. Utilizaremos por convención las letras en mayúscula para referirnos al conjunto de actividades (e.g., $A,B \subseteq A$), mientras que para las actividades individuales minúscula (e.g., $a, b, c, \ldots \in A$). La salida del algoritmo α es una red de Petri. $\alpha(L) = (N,M)$. El objetivo es descubrir redes de workflow. Por lo tanto, podemos omitir el marcado inicial y escribir $\alpha(L) = N$ y el valor inicial de marcado es implícito $M = [i]$).

El algoritmo α escanea el log de eventos para descubrir patrones particulares. Por ejemplo, si la actividad *a* es seguida por la actividad *b* pero *b* nunca es seguida por *a*, entonces se asume que hay una dependencia casual entre *a* y *b*. Para reflejar esa dependencia, la red de Petri correspondiente debería mostrar la conexión entre *a* y *b*. Se distinguen cuatro relaciones de orden basadas en el log que tienen el objetivo de capturar los patrones relevantes del log.

1.4.1 Tendencia representacional

El descubrimiento de procesos está restringido, por definición, por el poder expresivo del lenguaje de salida del modelo, es decir de la tendencia representacional.

La tendencia representacional ayuda a limitar el espacio de búsqueda entre los modelos candidatos. Esto hace que los algoritmos de descubrimiento sean más eficientes. Sin embargo, también puede ser usado para darle preferencia a tipos de modelos particulares. Parece que la existencia de aproximaciones puede beneficiar la selección de una tendencia representacional más adecuada. Por ejemplo el algoritmo α puede producir modelos que tengan deadlocks o livelocks. Aquí sería buena idea tener una tendencia representacional que limite el espacio de búsqueda a solo modelos con *sonido*. (Es decir modelos sin deadlock ni livelock). Por desgracia, esto solo puede ser realizado limitando severamente la expresividad de un lenguaje de moldeamiento o utilizando técnicas de análisis que consuman más tiempo. [4]

1.4.2 Ruido e Incompletitud

Para descubrir un modelo de proceso adecuado, se asume que el log de eventos contiene comportamiento de ejemplos representativo. Sin embargo, hay dos fenómenos que puede hacer al log de eventos menos representativos para el proceso estudiado:

- Ruido: el log de eventos contiene comportamiento raro y no frecuente que no es representativo del comportamiento típico del proceso. Este ruido se puede dar por la carga de eventos incorrectos, o errores que puede ocurrir mientras se almacenan los eventos.
- Incompletitud: el log de eventos contiene muy pocos eventos como para permitir el descubrimiento de las estructuras de control de flujo subyacente. [4]

1.4.3 Cuatro criterios de calidad

La completitud y el ruido se refieren a cualidades del log de eventos y no dicen mucho sobre la calidad del modelo descubierto. Determinar la calidad de un resultado de *Process Mining* es difícil y se caracteriza por varias dimensiones. Ahora nos referiremos a cuatro dimensiones de calidad: *fitness*, *simplicidad*, *precisión* y *generalización*, mencionados previamente. En la figura 5 se ve un alto nivel de caracterización de las cuatro dimensiones de calidad. Un modelo con buen *fittnes* permite el comportamiento observado en el log de eventos. Un modelo tiene un *fitness* perfecto si todas sus traces en el log se pueden reproducir en el modelo desde principio a fin. Hay varias maneras de definir fitness. Se puede definir a nivel de caso, por ejemplo, la fracción de traces en el log que puede representarse completamente. También se puede definir a nivel de evento, por ejemplo, la fracción de eventos en el log que son realmente posibles de acuerdo al modelo. Cuando definimos fitness, se deben tomar muchas decisiones de diseño. Por ejemplo cual es la penalización si un paso debe ser salteado y cuál es la penalización si los tolkens permanecen en la red de workflow luego de la reproducción.

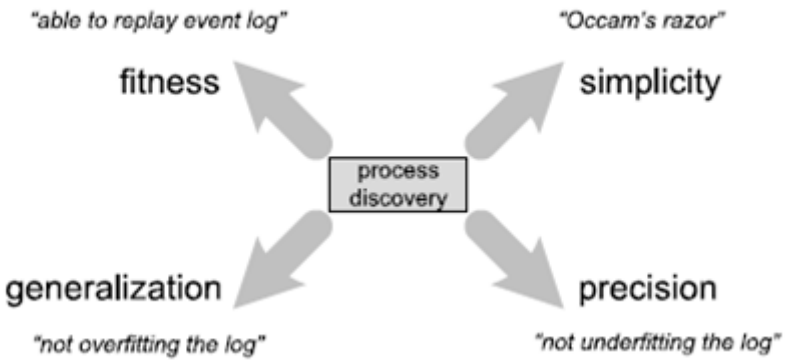


Figura 5 Balance de las cuatro dimensiones de calidad [4]

La dimensión de simplicidad se refiere a la navaja de Occams (Anexo 1). En el contexto de descubrimiento de procesos, esto quiere decir que el modelo más simple que puede explicar el comportamiento observado en el log es el mejor modelo. La complejidad del modelo se puede definir como el número de nodos y arcos en el grafo subyacente. También se pueden utilizar técnicas más sofisticadas como métricas que tome la estructura y la entropía del modelo tomado en cuenta.

Fitness y simplicidad solas no son suficientes. El modelo de la figura 6 tiene en cuenta estas dos métricas, con un fitness perfecto pero no es preciso y es muy generalizado.

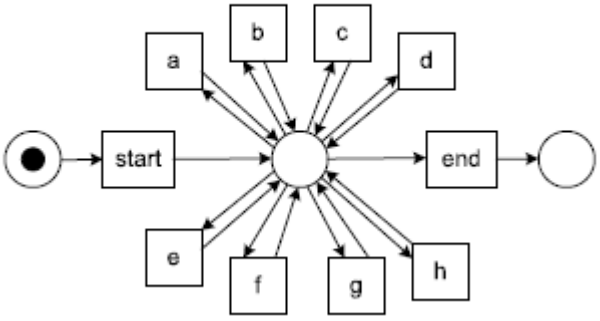


Figura 6 El modelo de "flor" de red de Petri permitiendo cualquier log que contenga las actividades {a,b, …, h} [4]

Los modelos extremos como el modelo flor en el cual todo es posible y el modelo enumerativo donde solo el log es posible muestra la necesidad de las otras dos dimensiones. Un modelo es preciso si no permite "demasiado" comportamiento. Claramente el "modelo flor" no tiene precisión. Un modelo que no es precisión tiene "underfitting". Underfitting es el problema en el que el modelo sobre generaliza el comportamiento de ejemplo del log, es decir, el modelo permite comportamiento muy diferentes de los que se ven en el log.

Un modelo debe generalizar y no restringir el comportamiento a los ejemplos que se vieron en el log (como el modelo enumerativo). Un modelo que no generaliza tiene *overfitting*. El *overfitting* es un problema en el que se generan modelos muy específicos mientras que es obvio que el log solo contiene comportamiento de ejemplo, es decir, el

modelo explica el log de ejemplo particular, pero un próximo log de ejemplo del mismo proceso puede producir un modelo de proceso completamente diferente.

Los algoritmos de *Process Mining* tienen que encontrar un balance entre el overfitting y el underfitting. Un modelo tiene overfitting si no generaliza y solo permite el comportamiento exacto almacenado en el log. Esto quiere decir que la técnica de data Mining correspondiente asume una fuerte noción de completitud: "Si la secuencia no se encuentra en el log, entonces ésta no es posible". Un modelo con underfitting sobre generaliza las cosas que ve en el log de eventos, es decir, permite más comportamiento aunque no haya indicaciones en el log que sugieran el comportamiento adicional.

En la figura 7 se ilustran cuatro modelos alternativos para el mismo log con distintos valores en los cuatro criterios de calidad.

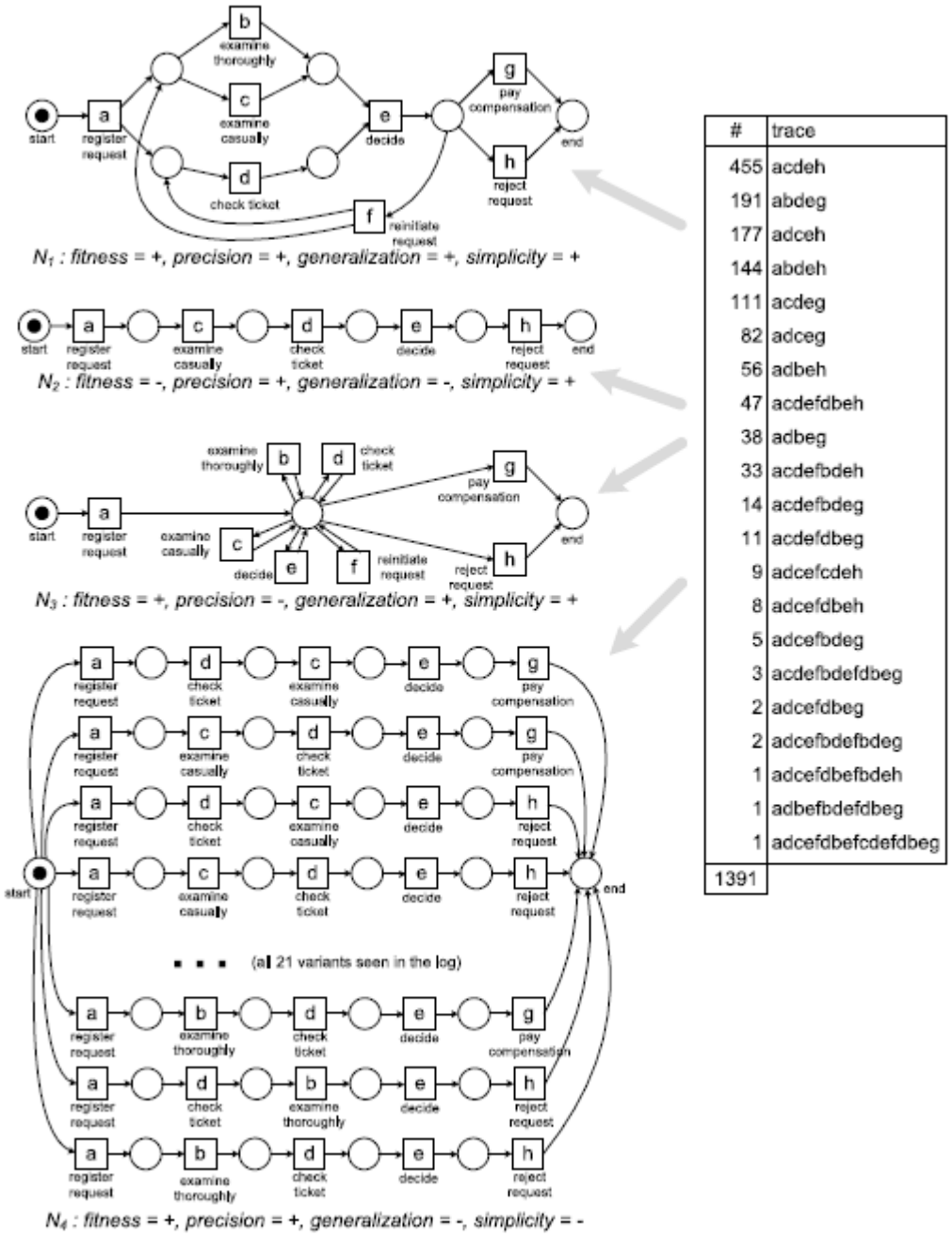


Figura 7 cuatro modelos alternativos para el mismo log. [4]

En la figura 8 se resumen todos los problemas mencionados en el contexto del algoritmo *α*. Cada punto representa un trace (es decir una secuencia de actividades) corresponden a uno o más casos en el log de eventos (recordar que múltiples casos pueden tener el mismo trace correspondiente) Un log de eventos contiene solo una fracción del comportamiento posible, es decir, los puntos deben ser visto como ejemplos de un conjunto más grande de comportamiento posible. Además suele interesar más el comportamiento frecuente y no todo el comportamiento posible, es decir, uno quiere abstraerse del ruido y

por lo tanto no todos los puntos tienen que ser relevantes para el modelo de procesos a construir.

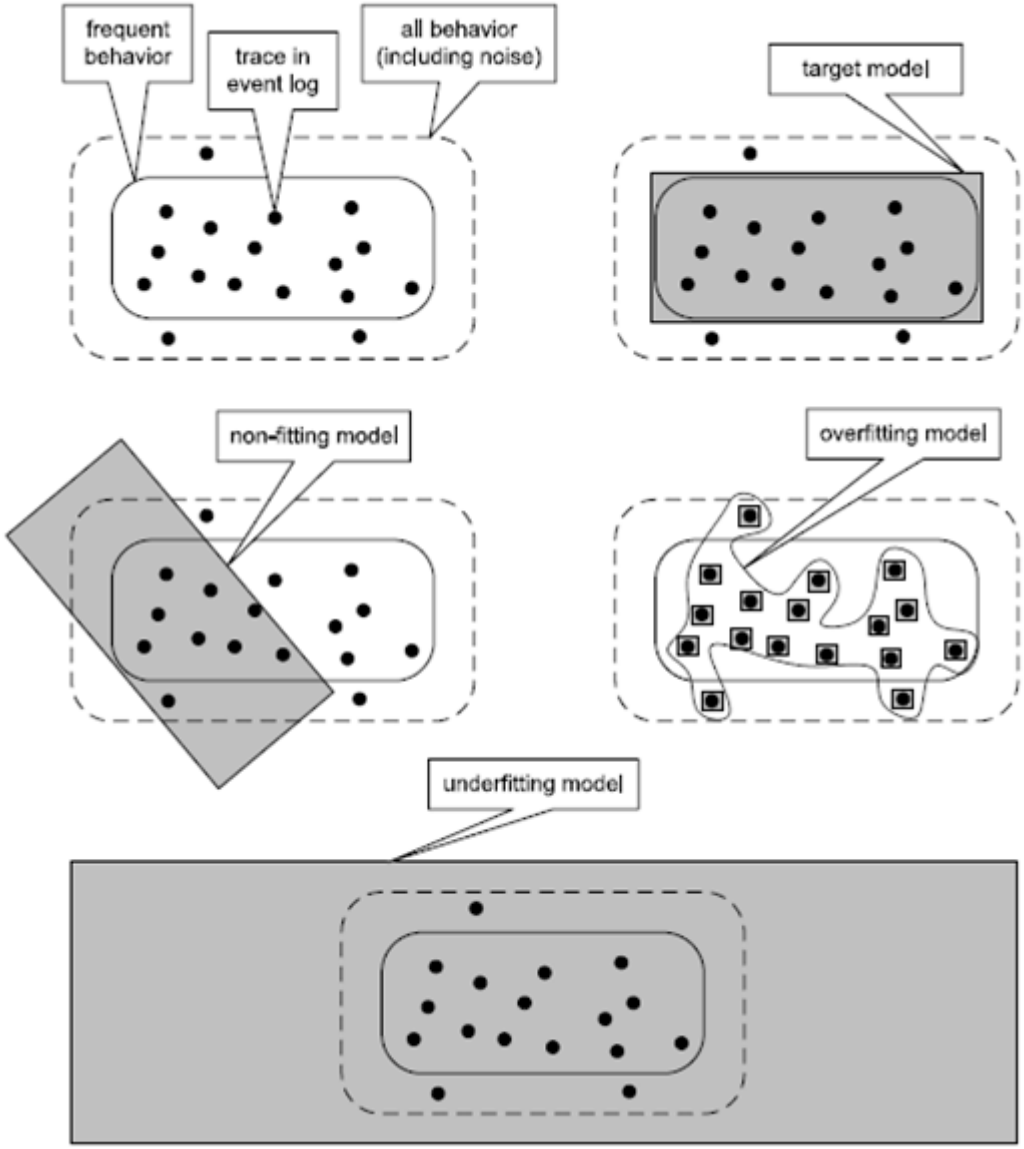


Figura 8 Desafíos a los que las técnicas de descubrimiento de procesos deben enfrentarse. [4]

Recordar que definimos al ruido como un comportamiento excepcional y poco frecuente. Es interesante analizar los comportamientos de ruido, sin embargo, cuando se construye el modelo de proceso total, la inclusión de comportamiento poco frecuente y excepcional lleva a diagramas complejos. La figura 9 distingue entre el comportamiento frecuente (el rectángulo con esquinas redondeadas) y todo el comportamiento (el rectángulo punteado), es decir el comportamiento de ruido y el normal.

En la figura 8 se muestra al modelo descubierto como el rectángulo sombrado. Estos modelos descubiertos están basados en los traces de ejemplo del log, es decir, los puntos negros. El modelo de proceso ideal permite al comportamiento coincidir con el

comportamiento frecuente. El modelo que no se adecua no es capaz de caracterizar bien al proceso ya que no es capaz de capturar los ejemplos del log de eventos para aprender el modelo. El modelo con overfitting no generaliza y solo es capaz de decir algo sobre los ejemplos en el log de eventos. Los ejemplos nuevos no podrán ser representados por el modelo. El modelo con underfitting no tiene precisión y permite comportamiento que nunca se vería en el procesos. [4]

El algoritmo α no está capacitado para extraer un modelo simple, sin underfitting, overfitting y que no sea adecue. Por eso se requieren de algoritmos de descubrimiento de procesos más avanzados como Heuristic Mining, minería basada en regiones, etc. Estas técnicas se verán con detalle en el anexo 3. Continuaremos con las técnicas de chequeo de concordancia.

1.5 Chequeo de concordancia

Las técnicas de chequeo de concordancia son técnicas del tipo *Replay* que utiliza tanto el log de eventos como el modelo de proceso como entrada, es decir la historia se vuelve a reproducir utilizando el modelo para analizar distintos fenómenos. Replay se puede utilizar para descubrir cuellos de botellas y análisis de decisiones. Para el chequeo de concordancia ya se posee el modelo de proceso y el log de eventos. El modelo puede haberse construido a mano o puede haber sido descubierto. El chequeo de concordancia relaciona eventos en el log de eventos con actividades en el modelo de proceso y las compara. El objetivo es encontrar las cosas en común y las discrepancias entre el comportamiento modelado y el comportamiento observado detectando desviaciones indeseables que provocan ineficiencias. . Además las técnicas de chequeo de concordancia pueden utilizarse también para medir la performance de los algoritmos de descubrimiento de procesos y para reparar modelos que no estén alineados con la realidad. La idea principal de chequeo de concordancia se ilustra en la figura 9.

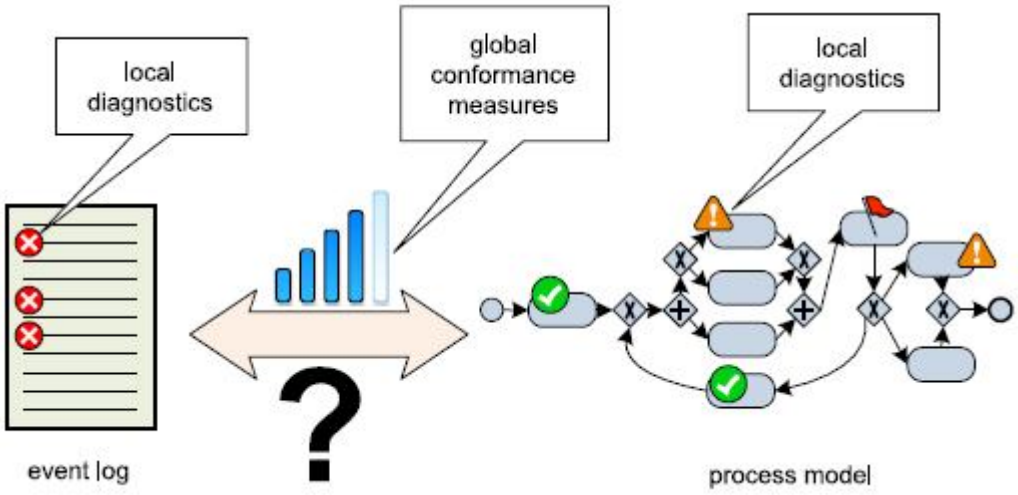


Figura 9 Chequeo de concordancia:. Las medidas globales de chequeo de concordancia cuantifican la concordancia total del modelo y el log. Se realizan diagnósticos locales sobresaltando los nodos del modelo donde el modelo y el log no concuerdan. [4]

El comportamiento de un modelo de proceso y el comportamiento almacenado en un log de eventos se comparar para encontrar cosas en común y discrepancias. Ese análisis resulta en

medidas de concordancia globales (por ejemplo 85% de los casos en el log de eventos se pueden reproducir con el modelo) y diagnóstico local (por ejemplo la actividad *x* fue ejecutada 15 veces a pesar de que no era permitido de acuerdo al modelo). Esta interpretación de no concordancia depende del propósito del modelo. Si el modelo intenta ser descriptivo, entonces las discrepancias entre el modelo y el log indican que el modelo necesita ser mejorado para adecuarse mejor a la realidad. Si el modelo es normativo, entonces las discrepancias encontradas puede exponer desviaciones indeseables, es decir, la necesidad de un mejor control de los procesos. Otras discrepancias pueden revelar desviaciones deseables. Por ejemplo, los trabajadores pueden derivarse a servir clientes mejor o a manejar circunstancias que no fueron previstas por el modelo de proceso. De hecho, la flexibilidad con la no concordancia por lo general correlaciona positivamente. Cuando se hace el chequeo de concordancia es importante ver las desviaciones desde dos ángulos:

- El modelo está mal y no refleja la realidad (Cómo mejorar un modelo?)
- Los casos se desvían con respecto al modelo y se requieren acciones correctivas (Cómo mejorar el control para hacer cumplir una mejor concordancia?)

Las técnicas de chequeo de concordancia deben contemplar estos dos puntos de vista.

El objetivo del alineamiento de negocios es asegurarse que la información de los sistemas y los procesos de negocio reales estén bien alineados. Las personas deben trabajar en conjunto con los sistemas de información antes que trabajar a su espalda para realizar las cosas.

Process Mining puede mejorar el alineamiento entre los sistemas de información, procesos de negocios y la organización. Mediante el análisis de los procesos reales y las discrepancias diagnosticadas, nuevos puntos de vista se pueden reunir mostrando cómo mejorar el soporte por sistemas de información.

El término auditoría se refiere a la evaluación de las organizaciones y sus procesos. La auditoría se realiza para comprobar la validez y la confianza de la información sobre estas organizaciones y los procesos asociados. Este procedimiento se realiza para chequear si los procesos de negocio se ejecutan dentro de ciertos límites seteados por los administradores, gobernadores y otros stakeholders. Por ejemplo, las reglas específicas pueden ser establecidas por la ley o políticas de compañías y el auditor se debe encargar de chequear si estas reglas se cumplen o no. La violación de esas reglas puede indicar fraude, mala práctica, riesgos e ineficiencias.

Hoy en día con el nivel de detalle que se tiene de los procesos almacenados en un log de eventos ya no debería ser aceptable chequear solo un pequeño conjuntos de ejemplos fuera de línea, en cambio todos los eventos en un proceso de negocio pueden ser evaluados y esto se puede hacer mientras el procesos siga corriendo. La capacidad de registrar datos y las técnicas avanzadas de *Process Mining* permite nuevas formas de auditoría, a través de *Process Mining*. [4]

En el anexo 3 se puede encontrar más detalle de las técnicas de chequeo de concordancia. Continuaremos con las técnicas de las perspectivas adicionales.

1.6 Minería de perspectivas adicionales

Mientras el foco principal del descubrimiento de procesos es en la perspectiva de control de flujo, el log de eventos puede contener información valiosa relacionada con otras perspectivas como la perspectiva organizacional, la perspectiva de caso, y la perspectiva de tiempo. La perspectiva organizacional se puede usar para obtener puntos de vistas de patrones de trabajo típicos, estructuras organizacionales y redes de trabajo social. Los timestamps y la frecuencia de las actividades se pueden utilizar para detectar cuellos de botella y diagnosticar otros problemas relacionados con la performance. Los datos de los casos se pueden usar para tener mejor entendimiento en la toma de decisiones y analizar diferencias sobre los casos. Además, las diferentes perspectivas se pueden mezclar en un único modelo que provea una vista integrada del proceso. Ese modelo integrado se puede utilizar para análisis "what if" utilizando simulación.

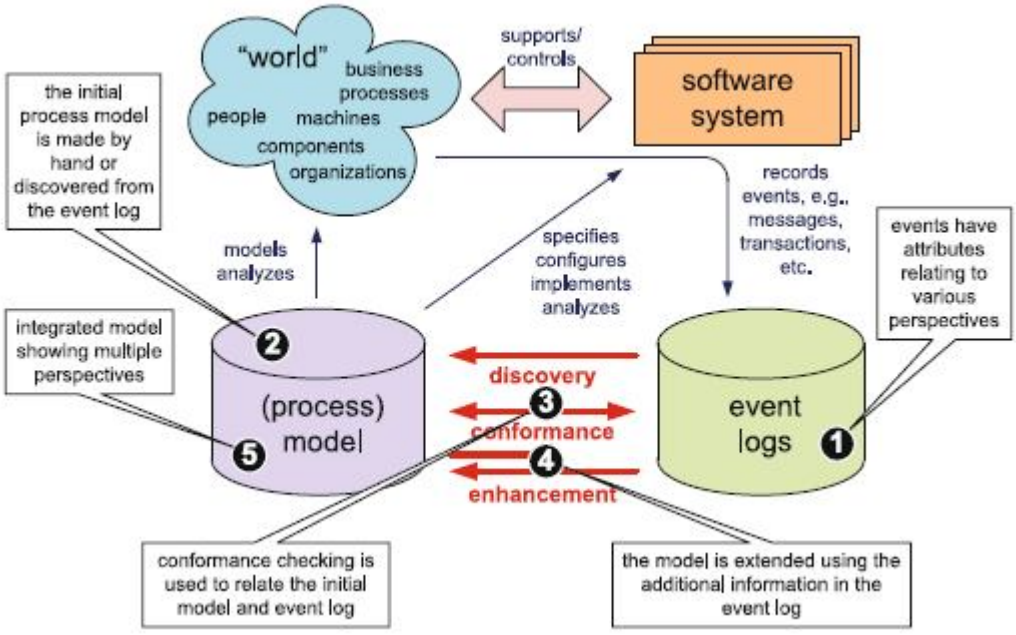


Figura 10 La perspectiva organizacional, de caso y de tiempo se pueden agregar al modelo de control de flujo original utilizando los atributos del log de eventos [4]

En la figura 10 se muestra un escenario típico. El punto de entrada es un log de eventos y un modelo de proceso inicial. El modelo de proceso se puede haber construido manualmente o por una técnica de process Discovery. Lo importante es que el log de eventos y el proceso estén conectados. Las aproximaciones de replay en el contexto de chequeo de concordancia se pueden usar para acoplar fuertemente al modelo y el log. Las instancias de actividad descubiertas durante la reproducción conectan las actividades modeladas a los eventos almacenados. De esta forma los atributos de eventos (recursos, timestamps, costos, etc.) se pueden usar para extender el modelo inicial. Por ejemplo, la información sobre servicios o tiempos de espera extraídos del log de eventos se pueden agregar al modelo. Luego de agregar las diferentes perspectivas, se obtiene un modelo de proceso integrado.
En la figura 10 se listan los tres tipos de *Process Mining*: descubrimiento, concordancia y mejora. Nos focalizamos en la mejora para extender y mejorar un modelo de proceso

existente utilizando información sobre el proceso actual almacenado en el log de eventos. Hay dos tipos de mejoras: reparación y extensión. Por medio de la extensión agregamos una nueva perspectiva al modelo de proceso mediante la correlación cruzada con el log. [4]

El primer paso en cualquier proyecto de *Process Mining* es obtener conocimiento del proceso y de los datos en el log de eventos. El llamado *dotted chart* provee una vista de helicóptero del proceso. En un *dotted chart*, cada evento se describe como un punto en un plano de dos dimensiones como se muestra en la figura 11 El eje horizontal representa el tiempo del el evento. El eje vertical representa la clase del evento. Para determinar la clase de un evento usamos el clasificador. Un clasificador es una función que mapea los atributos de u evento en una etiqueta, es la clase del evento. Un ejemplo de un clasificador es $e = \#_{case}(e)$, es decir el id del caso del evento. Otro ejemplo pueden ser $e = \#_{activity}(e)$ (el nombre de la actividad que se está ejecutando) y $e = \#_{resource}(e)$ (el recurso que dispara el evento).

El *dotted chart* se puede ver como un ejemplo de una técnica de análisis visual. La analítica visual aprovecha las capacidades humanas de identificar patrones y tendencias visualmente en conjuntos de datos grandes. [4]

Para más información detallada del *dotted chart* ver el anexo 3.

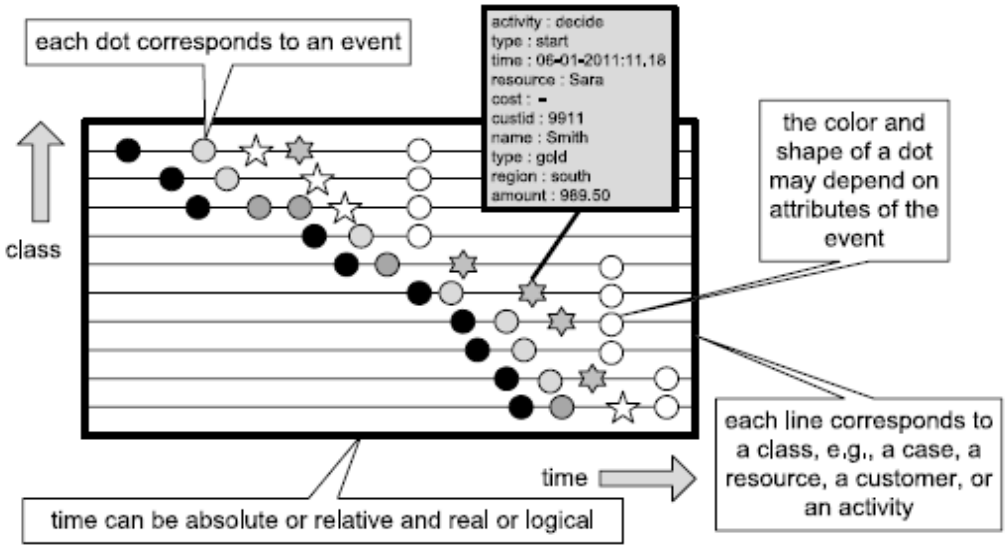


Figura 11 *Dotted chart*: los eventos se visualizan como puntos. La posición, el color y la forma dependen de los atributos del evento correspondiente. [4]

1.6.1 Minería organizacional

La minería organizacional se focaliza en la perspectiva organizacional. El punto de entrada para la minería organizacional es el atributo típico #resource(e) que se encuentra presente en la mayoría de los log de eventos. Utilizando esta información hay técnicas para aprender más sobre las personas, maquinas, estructuras organizacionales (roles y departamentos), distribución de trabajo y patrones de trabajo.

Análisis de redes de trabajo social

La sociometría, también conocida como sociografía, se refiere a métodos que presentan datos en relaciones interpersonales en forma de grafo o de matriz.

Para las técnicas de *Process Mining* utilizaremos redes sociales de trabajo como los que se ven en la figura 12. Los nodos en una red de trabajo corresponden a entidades organizacionales. Muchas veces hay una correspondencia de uno a uno entre los recursos encontrados en el log y las entidades organizacionales (es decir nodos). Los nodos en una red de trabajo social pueden corresponder a entidades organizacionales agregadas como roles, grupos, y departamentos. Los arcos en una red de trabajo social corresponden a relaciones entre estas entidades organizacionales. Los arcos y nodos pueden tener pesos. El peso de un arco o de un nodo indica su importancia. La interpretación de la importancia depende de la red de trabajo social. Muchas veces se asocia la distancia para referirse a la inversa de un peso, un arco con un peso muy alto indica que la distancia es corta. [4]

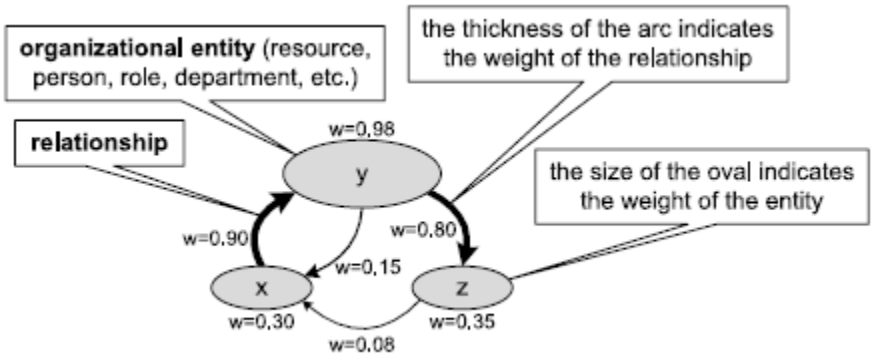


Figura 12 Un red de trabajo social consiste de nodos representando las entidades organizacionales, y arcos representando las relaciones. Tanto los nodos como los arcos pueden tener pesos indicados por "*w*=.." y el tamaño de la forma. [4]

Hay una gran variedad de métricas para analizar la red de trabajo social: centralización, cercanía, e intermediación. Nociones como centralización analizan la posición de una entidad organizacional, o persona en toda la red de trabajo social. Para más información sobre las redes de trabajo social ver anexo 3.

Analizando el comportamiento de los recursos

Las actividades, las entidades organizacionales y los recursos se pueden relacionar. Desde que los eventos en el log se refieren a actividades y recursos (e indirectamente a las entidades organizacionales), se pueden extraer del log mediciones de performance y se pueden proyectar en los modelos. Por ejemplo, se pueden proyectar frecuencias en las actividades, entidades organizacionales y recursos. Se puede ver cuántas veces un recurso ejecuto alguna actividad y de la misma manera saber cuántas veces fue usada una entidad organizacional.

Si el log de eventos contiene alta calidad de información incluyendo timestamps y tipos transaccionales, se puede analizar en detalle el comportamiento de los recursos. [4]

1.6.2 Tiempo y probabilidades

La perspectiva tiempo se concibe con la sincronización y la frecuencia de eventos. En la mayoría de los log de eventos hay un timestamp *(#time(e)*). La granularidad de estos timestamps puede variar. En algunos log solo se da información de la fecha, otros logs tienen información de los timestamps con precisión de milisegundos. La presencia de timestamps posibilita el descubrimiento de cuellos de botella, el análisis de niveles de servicios, el monitoreo de la utilización de los recursos y la predicción del tiempo que procesamiento de los casos corridos.

Todos los diagnósticos que se pueden hacer sobre el tiempo y probabilidades son posibles solo porque los eventos en el log se acoplan a elementos del modelo por medio de replay.

Luego de Replay para cada lugar una colección de visitas de tokens se registra. Cada visita de token tiene un tiempo de comienzo y de fin. Por lo tanto se puede derivar un conjunto múltiple de duraciones. Para un log de eventos grande ese conjunto múltiple contendrá cientos de elementos. Por lo que es posible ajustar una distribución y computar un estándar de estadísticas como, desviación estándar, máxima y mínima. Lo mismo sucede con instancias de actividad. Cada instancia de actividad tiene un tiempo de inicio y de fin. De esta forma se puede derivar un conjunto múltiple de tiempos de servicios. También de aquí se puede computar estadísticas estándar. Las estadísticas también se pueden computar para tiempos de espera. Es posible computar intervalos confidentes para derivar declaraciones como "el 90% de los intervalos confidentes para el tiempo medio de espera para la actividad *x* esta entre 40 y 50 minutos".

Tabla 2 Representación compacta de un log de eventos sobresaltando los timestamps, se utilizan timestamps artificiales para simplificar la representación del enfoque de replay basado en el tiempo. [4]

Case id	Trace
1	$\langle a^{12}_{start}, a^{19}_{complete}, b^{25}_{start}, d^{26}_{start}, b^{32}_{complete}, d^{33}_{complete}, e^{35}_{start}, e^{40}_{complete}, h^{50}_{start}, h^{54}_{complete}\rangle$
2	$\langle a^{17}_{start}, a^{23}_{complete}, d^{28}_{start}, c^{30}_{start}, d^{32}_{complete}, c^{38}_{complete}, e^{50}_{start}, e^{59}_{complete}, g^{70}_{start}, g^{73}_{complete}\rangle$
3	$\langle a^{25}_{start}, a^{30}_{complete}, c^{32}_{start}, c^{35}_{complete}, d^{35}_{start}, d^{40}_{complete}, e^{45}_{start}, e^{50}_{complete}, f^{50}_{start}, f^{55}_{complete}, b^{60}_{start}, d^{62}_{start}, b^{65}_{complete}, d^{67}_{complete}, e^{80}_{start}, e^{87}_{complete}, g^{90}_{start}, g^{98}_{complete}\rangle$
...	...

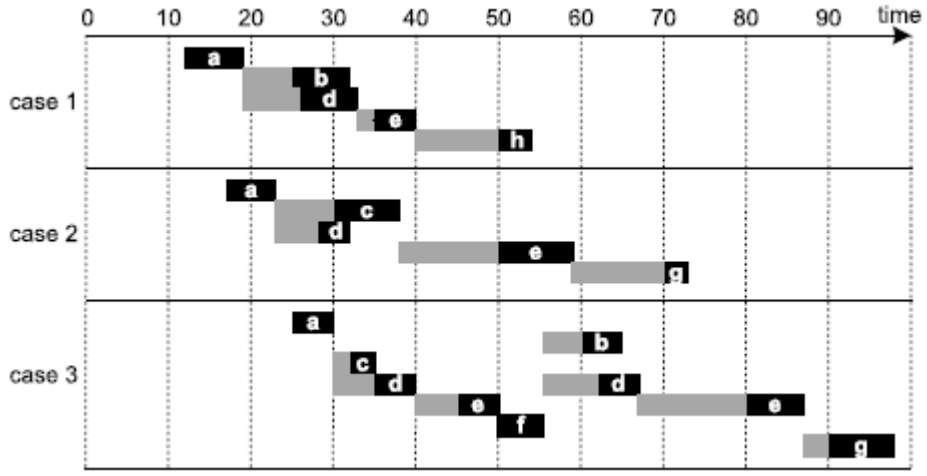


Figura 13 Línea de tiempo mostrando las instancias de actividades de los tres primeros casos. [4]

En la tabla 5 se muestra un fragmento de un log de eventos grande sobresaltando el rol de los timestamps. Para simplificar utilizamos timestamps de dos dígitos. Además asumimos que cada evento tiene un evento de comienzo y otro de finalización.

La figura 13 demuestra que la reproducción (replay) se puede usar para proveer información relacionada con la performance como:

- visualización de tiempos de espera y servicio: Las estadísticas como el promedio de tiempos de espera para una actividad se puede proyectar dentro de un modelo de proceso. Las actividades con una alta variación en el tiempo de servicios se puede sobresaltar en el modelo.
- Detección y análisis de cuellos de botella: el conjunto múltiple de duraciones adjunto a cada lugar puede ser utilizado para descubrir y analizar cuellos de botella. En los lugares donde más tiempo se lleva pueden sobresaltar. Además los casos que toman un largo tiempo en un lugar particular se pueden investigar más. El sublog de los casos demorados se puede analizar por separado para encontrar causas de la demora.
- Tiempo de flujo y análisis *SLA(Server Level Agreements)*: Se puede calcular el tiempo de flujo de trabajo total. Uno también puede señalar dos puntos arbitrarios en el proceso ya sea *x* y *z*, y computar cuantas veces un caso fluye desde *x* a *z*. El conjunto múltiple de duraciones va desde *x* a *y* se puede usar para calcular todo tipo de estadísticas , como ser, el promedio de tiempo de flujo entre *x* e *z* o la fracción de

los casos que tomas más de alguna norma presente. Esto se puede usar para monitorear Acuerdos a nivel de servicios (Server Level Agreements SLAs). Por ejemplo, es posible que sea un acuerdo contractual que el 90% de los casos *z* sea ejecutado dentro de las 48 horas después de que se complete *x*. La no concordancia con esos SLA se puede sobresaltar en el modelo.

- Análisis de frecuencia y utilización: cuando se reproduce el modelo (replay) se recolectan tiempos y frecuencias. Estas pueden ser usadas para mostrar las probabilidades de ruteo en el modelo. Por ejemplo luego de ejecutar *e* se elige entre seguir con *f*, *g* o *h*. Mediante el análisis de frecuencias, se puede indicar en el modelo que el 56% de las elecciones, *e* es seguida por *f*, el 20% *g* es elegida, y en el 24% se elige *h.* Mediante la combinación de frecuencias y promedios de tiempos de servicios, se puede calcular la utilización de los recursos.

Si el log de eventos tiene eventos como *asignado*, *suspendido*, *retomado*, *salteo manual*, *caso abortado*, se pueden recolectar más estadísticas durante la reproducción. Por ejemplo si los eventos de comienzo se proceden de los eventos de asignación, es posible analizar cuanto se toma en comenzar a ejecutar una actividad después de haber sido asignada a un recurso específico.
Si el log de eventos no tiene información transaccional es decir el atributo *#trans(e)* no se encuentra en el log, las actividades se suponen atómicas, sin embargo, todavía se puede analizar el tiempo que pasa entre esas actividades atómicas. Además se puede aplicar heurística para "adivinar" la duración de las instancias de actividad. [4]

1.6.3 Minería de decisión

La perspectiva de casos se focaliza en las propiedades de las distintas instancias de proceso. Cada caso se caracteriza por sus atributos, los atributos de sus eventos, el camino que se tomó e información de la performance (ejemplo tiempos de flujos).
Primero nos focalizamos en la influencia de atributos de casos y eventos en el ruteo de los casos.
La minería de decisión tiene como objetivo encontrar reglas explicando tales decisiones en términos de las características del caso. Se puede utilizar una técnica de clasificación como aprendizaje de árboles de decisión para descubrir ciertas reglas (visto en el anexo 1). Recordar que la entrada para el aprendizaje de árboles de decisión es una tabla donde cada fila lista una variable de respuesta categórica (ejemplo la actividad elegida) y múltiples variables predictivas (ejemplo propiedades del cliente). El árbol de decisión trata de explicar las variables de respuesta en término de las variables predictivas.

type	region	amount	activity
gold	south	987.30	z
silver	north	178.70	z
gold	south	211,50	y
silver	west	587.70	z
silver	east	224.70	z
silver	south	278,50	z
gold	north	488.50	y
silver	west	443.20	z
silver	south	673,70	z
gold	west	413.50	y
silver	south	687.70	z
gold	south	987,30	z
silver	north	378.80	z
gold	south	314,50	y
silver	north	537,70	z
silver	west	158.70	z
gold	east	344,50	y
...	...	...	...

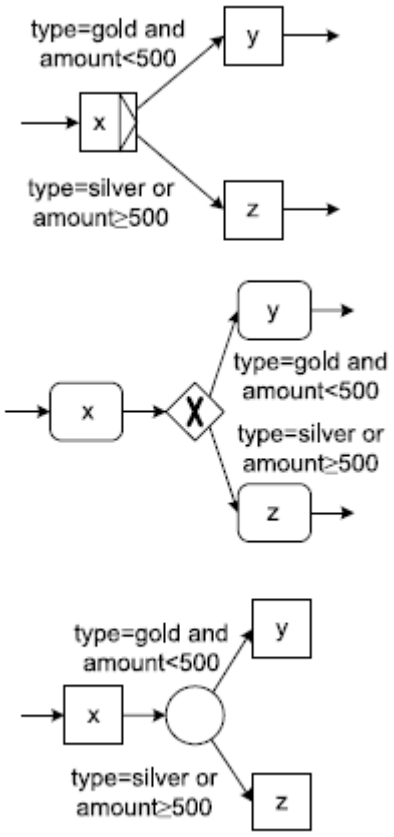


Figura 14 Minería de decisión utilizando atributos de caso y de evento, se aprende una regla para la decisión XOR. El resultado se muestra en diferentes notaciones YAWL (arriba), BPMN (medio) Red de Petri (abajo) [4]

Por ejemplo, como se ve en la figura 14, utilizando 3 notaciones distintas (YAWL, BPMN, y redes de Petri) se representa una elección: la actividad *x* se sigue por la actividad *y* o por la actividad *z*. Hay tres variables predictivas (tipo, región y monto) y una variable de respuesta (la actividad). Las variables tipo, región y actividad son categóricas mientras que la variable monto es numérica. Las variables predictivas corresponden al conocimiento obtenido del caso en el punto de tiempo donde la decisión se tiene que tomar. La variable de respuesta *actividad* se determina con un escaneo en el log de eventos. El log de eventos va a revelar si *x* fue seguido por *y* o por *z*. La tabla se puede usar para hacer árboles de decisiones y el árbol de decisión resultante se puede escribir en una regla. Basándose en la tabla, la clasificación mostrará que el valor de la variable respuesta es *y* si el cliente es gold y el monto es menor que 500$. De otra forma el valor de la variable respuesta es *z*. Las redes de petri no pueden expresar las separaciones OR-split y joins directamente. Sin embargo en lenguajes de alto nivel como BPMN y YAWL se puede expresar este comportamiento. Una actividad *x* puede ser seguida por *y*, o *z*, o por *y* y *z*. En este caso la variable respuesta también es categórica y se puede determinar con un escaneo en el log de eventos.

Puede haber loops en el modelo. Por lo tanto el mismo punto puede ser visitado múltiples veces en el mismo caso. Cada visita corresponde a una nueva columna en la tabla utilizada

por el algoritmo de árbol de decisiones. Utilizando replay, el resultado de la decisión (es decir la variable de respuesta) se puede identificar para cada fila.

En algunos casos es imposible la derivación de una regla de decisión razonable. La razón puede ser porque se dispone de pocos datos, o que las decisiones son aparentemente random o basadas en consideraciones que no se encuentran en el log de eventos. En esos casos, replay se puede usar para proveer una probabilidad para cada sucursal. Por lo tanto, ese punto de decisión se caracteriza por probabilidades en ved de reglas de decisión dependientes de datos.

El proceso se puede repetir para cada punto de decisión en el modelo de proceso. El resultado se puede usar para extender el modelo de proceso, para incorporar la perspectiva de caso.

Las técnicas de clasificación se pueden combinar con *Process Mining* una vez que el modelo de proceso y el log de eventos se conecten a través de las técnicas de replay.[4]

Luego de haber entrado en detalle en el marco conceptual de *Process Mining* y teniendo en cuenta sus objetivos y la potencialidad de las técnicas de *Process Mining* continuaremos con el marco conceptual de BPM y BPMS.

Capítulo 2: Marco conceptual de BPM y BPMS

El concepto clave de la administración de procesos de negocio (BPM) es la convergencia de la tecnología con las teorías de administración de procesos. Esta convergencia produce nuevos enfoques de diseño e implementación de procesos que posibilitan la llamada empresa de proceso. La empresa de proceso se organiza alrededor de procesos centrales que atraviesan los departamentos y líneas que dividen, las empresas miden y estandarizan estos procesos. Utilizando el enfoque y la tecnología de diseño de procesos de BPM (tecnología que llamamos Bussines Process Management (BPMSs)), las soluciones de BPM permiten a la empresa de proceso medir y estandarizar procesos y proveer procesos reusables que se puedan conectar. Este nuevo tipo de tecnología facilita la tarea de cambiar procesos de negocio mediante la separación de las aplicaciones subyacentes de los mismos.

BPM es entonces un enfoque sistémico estructurado para analizar, mejorar, controlar y administrar procesos con el objetivo de mejorar la calidad de los productos y servicios.[1][2][17] Es entonces cuando *Process Mining* entra en juego como tecnología de soporte para asistir a BPM.

En este capítulo se verá un marco conceptual de BPM y BPMS que nos ayudaran a abordar luego los enfoques de integración entre *Process Mining* y BPM.

2.1 Principios de la administración de procesos de negocios (BPM)

BPM tiene 4 principios básicos sobre los que se construye como disciplina, a continuación veremos en detalle dichos principios:

1. El principio básico de BPM es que **los procesos son bienes que producen valor para los clientes**. Un proceso tiene clientes internos o externos y son los receptores de los resultados creados por los procesos. Los individuos y las funciones no producen valor para los clientes. Pueden ser responsables de una parte del trabajo total, pero los clientes no perciben el valor de las funciones independientes. Un ejemplo es una función de ventas. Las personas de venta se pueden consideran a sí mismas generadores de ingresos, y pueden tener una visión exagerada de su importancia en la organización. Sin embargo, sin el servicio de clientes, cuentas, manufactura, y otras funciones, los clientes no recibirían ningún valor de la función de ventas. Los procesos son responsables del trabajo de principio a fin que devuelve un valor al cliente y es por eso que para las organizaciones analizarlos y optimizarlos requieren de una tecnología que se focalice en el proceso de inicio a fin, como *Process Mining*. Las organizaciones deben invertir en procesos como lo hacen en otros bienes. Esto no quiere decir que todos los procesos son el núcleo de una organización. Diferentes organizaciones tienen diferentes objetivos. Por lo tanto, los procesos principales difieren.
2. El segundo principio es que **los procesos deben ser administrados y mejorados continuamente**. Como los procesos son bienes, los procesos principales y los procesos que generan mayor valor para los clientes, deben ser administrados cuidadosamente. Un proceso administrado produce valor

consistente para los clientes y tiene la base para que el proceso sea mejorado. La administración de procesos implica la tarea de medir, monitorear, controlar y analizar los procesos de negocio. Estas tres tareas van de mano en mano. Medir los procesos de negocio provee información con respecto a esos procesos de negocio. La información de los procesos permite a las organizaciones predecir, reconocer y diagnosticar deficiencias del proceso y sugiere el camino para mejoras futuras. El monitoreo de proceso parecido al control de procesos estático (SPC) en la ingeniería industrial y la administración de calidad. Cuando los procesos se monitorean se pueden detectar las varianzas. Un proceso que tiene una alta variabilidad es un proceso que produce resultados inconsistentes para los clientes y la alta variabilidad indica que hay un problema en el proceso. Una vez que el proceso demuestra una alta variabilidad, debe haber un mecanismo permitiendo controlar al proceso. Este mecanismo puede agregar más recursos (personas, maquinaria, etc.), dando de baja el proceso si la situación es grave, o activando procesos alternativos. Analizar información de proceso es un paso esencial para identificar lo que los procesos necesitan mejorar, y cuales mejoras son las que traerán mayor valor al proceso.

3. El tercer principio es la **mejora continua de los proceso.** Este es un resultado natural de la administración de proceso. La mejora de proceso se facilita por la disponibilidad de información del proceso. La mejora de procesos no es una tarea fácil. El entorno de negocios por lo general indica que una organización debe mejorar para mantenerse competitiva. Los procesos de negocio son centrales para la creación de valor en una organización. Por lo tanto los procesos deben ser mejorados continuamente. Dentro del marco de BPM, BPR y otras metodologías de mejora de proceso (como Six Sigma) son herramientas que las organizaciones pueden utilizar para implementar la mejora de proceso. Focalizándose en la mejora continua, una organización se encuentra mejor preparada para enfrentar los cambios, que son constantes en la economía orientada al cliente. Esto ayuda a desarrollar una cultura corporativa que es orientada al proceso y que está lista para adaptase a los cambios.
4. El último principio es que **las tecnologías de la información (IT) son un activador esencial**. El foco en los proceso tiene sus orígenes en la disciplina de ingeniería industrial. Concebida originalmente para mejorar funciones de logística y manufactura, la ingeniería industrial (IE) se focaliza en procesos y utiliza estadísticas como herramienta de habilitación de monitoreo de varianzas de procesos. Se propuso el uso de las tecnologías de información (IT) como herramienta esencial para la nueva disciplina de ingeniería industrial. En la nueva ingeniería industrial, los procesos de negocio son el foco para la mejora e IT es la herramienta habilitadora clave. IT puede proveer información de procesos en tiempo real que es muy importante para BPM para cumplir sus tareas de monitoreo y control de procesos de negocio. No es factible medir el tiempo real de procesos de negocio sin un mecanismo de medición automatizado. Las tecnologías de información proveen este mecanismo. IT permite que BPM integre diseño de procesos, desarrollo e implementación.[17]

2.1.1 Utilización de tecnologías de información para administrar procesos

La adopción de BPM y técnicas de administración de procesos por las organizaciones ha originado un fuerte interés en el mundo tecnológico para desarrollar productos y soluciones para soportar BPM. La tecnología BPM ha madurado al punto donde la administración de procesos en tiempo real es posible. Los BPMS representan un quiebre en el uso e implementación de sistemas de información. La metodología de implementación de los sistemas tradicionales se focaliza en funciones y objetos. Los procesos se relegan al flujo de trabajo, que por lo general no recibe mayor atención durante la implementación. Los BPMS rompen con la mentalidad de diseño de objetos. Pone al proceso en un foco central de diseño de soluciones. Esto alinea a las soluciones de IT a estar más en línea con la realidad de proceso para las organizaciones de BPM. Además de las funciones de diseño, un BPMS, permite que una vez implementado se puedan medir, monitorear, controlar y analizar el tiempo real de los procesos. En pocas palabras, los BPMS sirven como el centro de control sobre las personas, aplicaciones de empresa, y datos. Como centro de control los BPMS reciben datos en tiempo real de todas las tareas que se realizan en los procesos que controla. Es posible el monitoreo con la información de proceso en tiempo real. BPMS puede detectar variaciones en el proceso, una vez que se detecta una variación, un BPMS tiene la capacidad de resolver esa variación. Las mediciones en el proceso de negocio proveen datos de proceso que se pueden analizar. El análisis de los datos de proceso es un paso esencial para identificar qué proceso debe mejorar y las mejoras a que área pueden traer mayor valor al proceso.[17] *Process Mining* podría asistir al BPMS en esta tarea de análisis.

2.2 Capacidades claves de un BPMS

Los BPMS son una clase de software que permite a las organizaciones crear soluciones de tecnología de la información centradas en el proceso. Que se centre en el proceso quiere decir que las soluciones de un BPMS permiten integrar personas, sistemas y datos. Las organizaciones que utilicen un BPMS para llevar a cabo el cambio de procesos de negocio habilitado por IT ganaran las siguientes capacidades:

1. Involucrarse mejor en el negocio en el diseño de soluciones de procesos de negocio habilitado por IT.
2. Habilidad de integrar personas y sistemas que participan en los procesos de negocio.
3. Habilidad de simular procesos de negocio para diseñar el proceso más óptimo para la implementación.
4. Habilidad de monitorear, controlar y mejorar procesos de negocio en tiempo real.
5. Habilidad de efectuar cambios en procesos de negocio existentes en tiempo real sin esfuerzo de conversión de proceso elaborado.

BPMS permite que los propietarios de los procesos de negocio se involucren directamente con la solución IT. El diseñador de procesos que provee un BPMS, es una herramienta que permite a los propietarios de los procesos de negocio o a los analistas de negocio diseñar los procesos de negocio en detalle. Los diseñadores de procesos de negocio pueden generar automáticamente el código que a veces se puede desplegar sin

ayuda del departamento de IT. En el caso en el que se requiera un desarrollo, el BPMS ya contiene el significado del proceso y las definiciones de la solución. Los desarrolladores de IT pueden utilizar el mismo proceso diseñado por el personal de negocios, y embeber la lógica al proceso utilizando el diseñador. Esto se puede realizar utilizando el lenguaje de script que viene empaquetado con el diseñador de procesos o la herramienta de desarrollo en un lenguaje de programación utilizado extensamente. Tanto el personal de negocio como el personal de IT trabajan desde el mismo diseño utilizando la misma herramienta. Esto reduce la brecha comunicacional entre IT y los negocios. [17]

2.2.1 Integración centrada en el proceso

La palabra proceso se ha utilizado ampliamente en IT. Sin embargo, no hay una definición para proceso. Los propulsores de la administración de documentos pueden ver a los proceso como flujos de documentos que soportan una transacción de negocio. Los propulsores de workflow ven al proceso como tareas que deben ser ejecutadas por humanos. Para prevenir la confusión que la palabra proceso causa en IT, nos referiremos al termino integración centrada en el proceso, lo que quiere decir la integración de personas, sistemas y datos. BPMS es la primer clase de software que ofrece una integración centrada en el proceso. Esto trasciende la capacidad de workflow tradicional que se encuentra disponible desde hace años. De manera similar a los sistemas de administración de workflow, BPMS ofrece capacidades de workflow que pueden generar listas de trabajo para participantes humanos en el proceso de negocio para que ejecuten sus tareas como se muestra en la Figura 59. La lista de trabajo se presenta a los participantes humanos en una página Web por medio de un navegador de internet. Estas listas de trabajo se puede basar en roles e integrarse con un portal de la empresa para empleados. Para completar tareas en esta lista de trabajo, el trabajo se presenta de mediante formularios Web simples y entendibles para que el usuario los complete. Los sistemas involucrados en los procesos de negocio también se integran por medio de las capacidades de integración de aplicaciones que ofrece el BPMS. La integración centrada en el proceso permite a BPMS conectar cada rol y tarea que se especifique en el diseñador de procesos de negocio.

La tecnología BPM permite a los analistas de negocio colaborar mejor con el personal de IT en la implementación de los proyectos. Las herramientas variadas que BPMS ofrece provee un nuevo paradigma para la implementación de soluciones. Las organizaciones ya no están atadas a los procesos de negocio atadas a las aplicaciones de negocio. Con un generador de workflow automático y un portal web, el workflow se puede desplegar fácilmente a través de múltiples aplicaciones, y por lo tanto integrar personal en los procesos de negocio. Los procesos se administran en un framework de procesos.

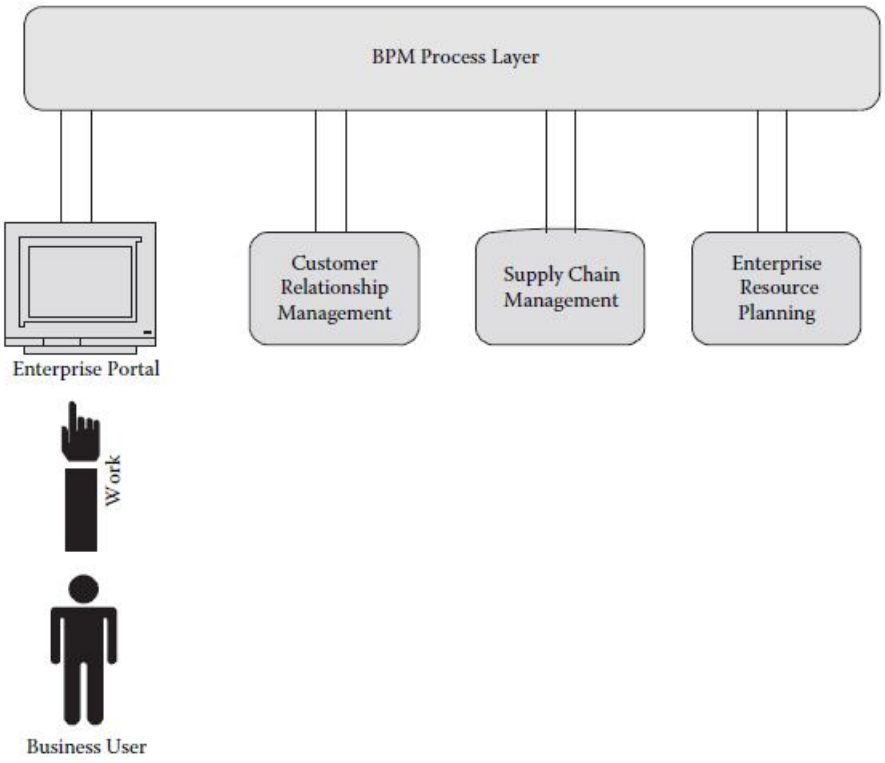


Figura 15 Integración de personal y sistemas por un BPMS [17]

En el diseñador de procesos de negocio, el analista puede especificar roles, tareas que deben realizarse por varios roles, y una secuencia que el proceso debe seguir. Los roles pueden ser para personal o para sistemas. Por lo general los proceso no siguen un flujo simple secuencia. Hay muchos escenarios excepcionales. El diseñador de procesos permite al analista configurar las condiciones y los flujos de estas excepciones y subprocesos. En el diseñador de procesos, el analista de procesos puede diseñar el proceso como le indica el entorno de negocio sin preocuparse por la funcionalidad de la aplicación. No es necesario que el analista de proceso entender el trabajo interno de las aplicaciones subyacentes. El analista solo debe entender la lógica y el flujo de las tareas para excepciones que deben ser manejadas por aplicaciones o participantes humanos. Una vez que la solución de proceso de negocio se implementó, el trabajo se presenta a los usuarios de negocio en una lista de trabajo por medio de la aplicación portal de la empresa. Los usuarios de negocio seleccionan un elemento de trabajo y se presenta un formulario Web para completarse. Cuando se completa el formulario, la instancia de proceso procede a la próxima actividad.[17]

2.2.2 Simulación de procesos

Para ayudar a los propietarios de los procesos de negocio y a los analistas de negocio en el diseño de proceso, BPMS provee la simulación de procesos y capacidades de modelado. Utilizando el diseñador de procesos de negocios del BPMS, se puede diseñar el proceso de negocio inicial y correr los diseños de procesos en un modo de simulación. El modo de simulación incluye distribuciones de probabilidad de tiempo para cada actividad en el proceso simulado. Se debe hacer un trabajo de preparación para determinar que algoritmo de simulación utilizar y que modelo de distribución probabilístico se adecua para

cada actividad en el proceso de negocio. Una vez que se colectan los datos, el simulador de procesos identifica que pasos son los cuellos de botella y cualquier otra debilidad en el diseño de proceso. Basado en los resultados de simulación el diseño de proceso inicial se puede mejorar iterativamente. [17]

2.2.3 Administración de procesos

BPMS sirve como sistema supervisor que supervisa el proceso de negocio una vez que la solución se implementó. El aspecto supervisor de un BPMS provee capacidades de monitoreo, control, y mejora de procesos de negocio. Como BPMS supervisa cada paso, ya sea manual o automático del proceso de negocio, y puede proveer información importante del proceso. El software de BPMS sirve como monitor de performance de los procesos. Se pueden obtener estadísticas como tiempo de ciclo promedio por transacción, el tiempo de espera antes que una tarea de proceso se ejecute por los participantes humanos y datos de costo. Las capacidades de monitoreo permiten que se les notifique a los administradores de procesos de los eventos que se encuentran fuera de lo común. [17]

2.2.4 Mejora de procesos en tiempo real

BPMS da a las organizaciones la capacidad de implementar mejoras de procesos en tiempo real sin demasiado esfuerzo de conversiones de procesos. El proceso de negocio original ya existe en el diseñador de procesos de negocio. Esto elimina la necesidad de reunir la información del proceso actual. Cuando se detectan cuellos de botella, se pueden incorporar las mejoras al proceso utilizando el diseñador de procesos de negocio. Luego de la implementar la solución del proceso mejorado, el BPMS permite que cualquier instancia comenzada con el proceso original finalice utilizando la definición del proceso original, y que las nuevas instancias de proceso se realicen utilizando la definición del proceso mejorado. Se pueden realizar mejoras en los procesos sin trastornar la salida del proceso.[17]

2.3 Introducción a la capa de proceso

La capa de procesos de negocio es donde reside el BPMS. La arquitectura tradicional, antes de la arquitectura con capa de proceso estaba compuesta por tres capas: la capa de base de datos, la capa de aplicación y la capa de presentación. En esta estructura, la base de datos es donde se almacenan los datos físicamente. La capa de aplicación contiene las aplicaciones de negocio y la lógica del proceso, y la capa de presentación es lo que el usuario ve. En la arquitectura empresarial de cuatro capaz, se incorpora la capa de proceso entre la capa de presentación y aplicación. Los BPMS ocupan su rol central en la capa de proceso. En la figura 60 se ilustra la arquitectura en tres capas y en cuatro capas.[17]

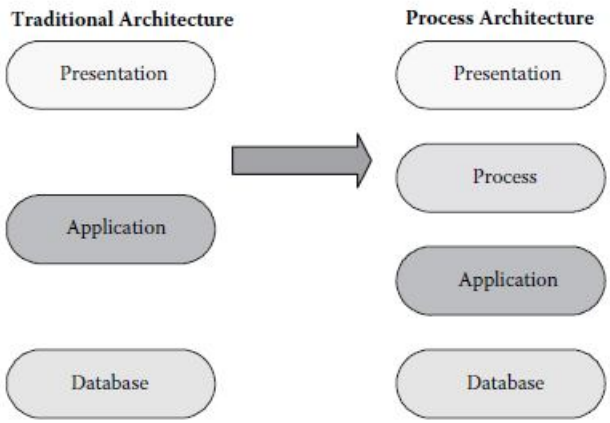


Figura 16 Arquitectura de tres y cuatro capas.[17]

2.3.1 Deficiencias de la interfaz punto a punto

La capa de proceso tiene como objetivo integrar datos y aplicaciones. En una arquitectura de tres capas, la integración con otras aplicaciones se hace punto a punto. Cuando una aplicación requiere de datos específicos de otra aplicación, se crea una interfaz para conectar las dos aplicaciones y trasferir los datos específicos. A medida que los requerimientos de negocio crecen y las aplicaciones se crean y se modifican, el número de aplicaciones creadas puede crecer enormemente. Hay varios problemas con el enfoque punto a punto con integración de aplicaciones. Primero, el número de interfaces requiere muchos programadores y un alto presupuesto para mantenerlos. Segundo, no es fácil imponer estándar común de datos para interfaces punto a punto. Tercero, cualquier cambio en el modelo de datos de una aplicación que se conecta por medio de interfaces, puede resultar en altos costos para actualizar las interfaces afectadas.[17]

2.3.2 Sistema de administración de procesos de negocio (BPMS) framework de integración de aplicaciones

Lo que la capa de procesos provee, por medio del BPMS, es el nuevo paradigma hacia un framework de integración de aplicaciones. Este framework viene con herramientas de desarrollo, conectividad a sistemas comercialmente disponibles, mapeo de datos y otras herramientas. Este framework de integración de aplicaciones permite a las corporaciones crear esquemas de datos para toda la empresa que todas las aplicaciones deben conformar para transferir datos. XML permite que los esquemas no sean tan rígidos y que sean más extensibles. El framework de integración de aplicaciones representa el eje de datos donde todas las aplicaciones transmiten y reciben información. Todas las aplicaciones se conectan a la capa de proceso por medio de conectores y adaptadores, que son software que encuentran comercialmente disponibles para permitir la conectividad con aplicaciones de negocio. El repositorio de datos en la capa de proceso permite a las aplicaciones que obtengan los datos necesarios para sus funciones sin ir directamente a las aplicaciones originarias. Esto elimina la necesidad de interfaces punto a punto y reduce drásticamente el número de conexiones de integración para mantener. En la figura 17 se muestra la integración de aplicaciones de la capa de proceso versus la integración de aplicaciones punto a punto. El resultado del framework de integración de aplicaciones en la capa de proceso es menos desarrollo, costo de mantenimiento y tiempo más rápido de despliegue.[17]

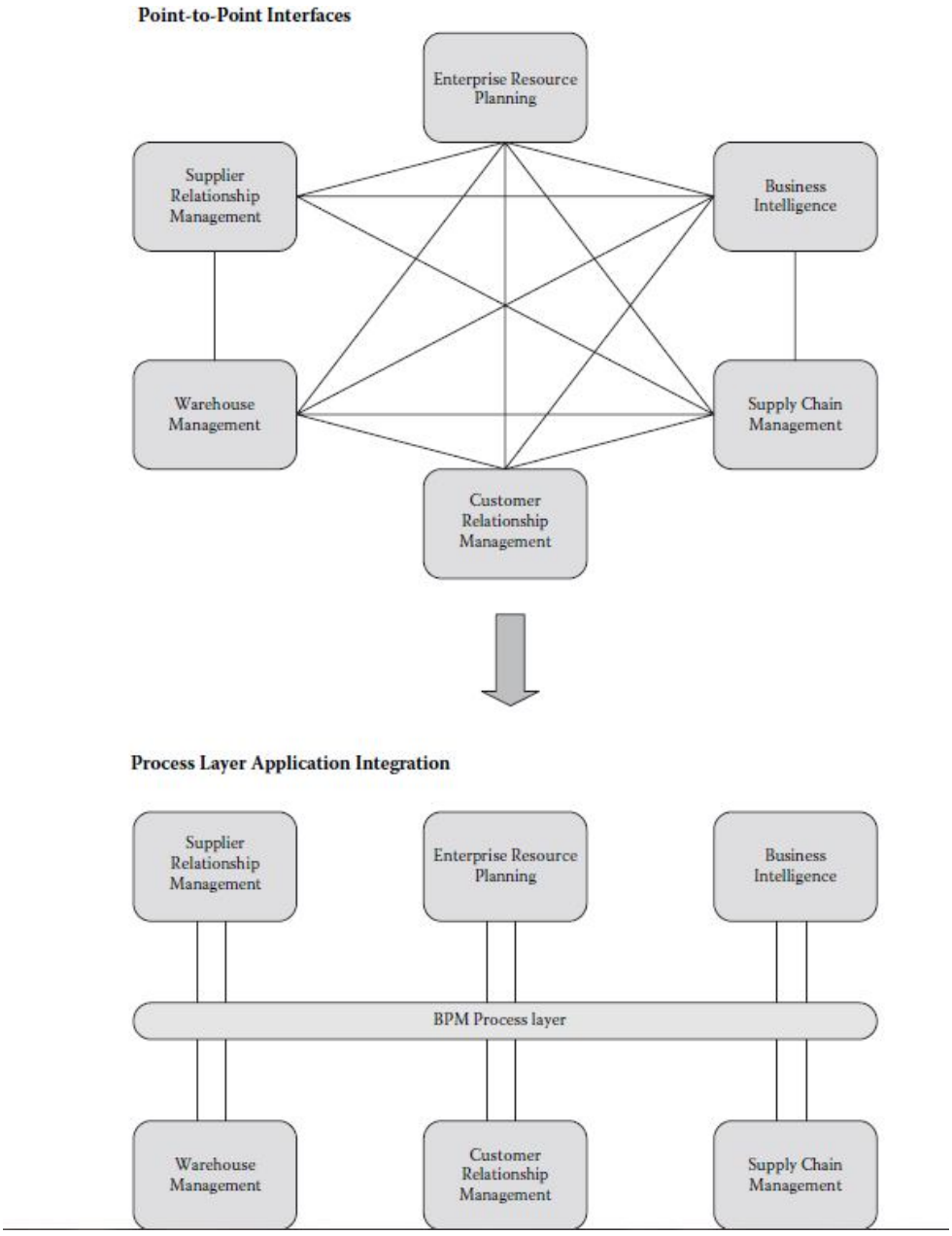


Figura 17 integración de aplicaciones de capa de procesos versus integración de aplicaciones punto a punto [17]

Luego de haber analizado los marcos conceptuales de *Process Mining* y BPMS y tener en claro los objetivos y beneficios de cada una de estas metodologías, nos encontramos en condiciones de encarar el enfoque para la utilización de *Process Mining* como medio para analizar los procesos de negocia que se encuentran implantados en un BPMS.

Capítulo 3 Análisis y propuesta de un enfoque sobre *Process Mining* como medio para analizar los procesos de negocio ya implantados en un BPMS

Como vimos en el capítulo 1 la mayoría de las herramientas de *Business Intelligence* (BI) utilizan datos de eventos para soportar la toma de decisiones. Bajo el paraguas de BI se encuentran otras tecnologías, como BAM (Monitoreo de actividades de negocio), CPM (Gestión del rendimiento corporativo), CPI (Mejora continua de procesos), and BPI (Inteligencia de procesos de negocio) que permiten realizar reportes y dashboards (tableros de mando), pero solo se focalizan en los datos y no en el proceso de inicio a fin.

Por otro lado los BPMS utilizan modelos de proceso para analizar procesos operacionales. Por lo general estos modelos se encuentran desconectados de los datos de eventos reales, y por lo tanto los resultados pueden ser no confiables ya que los mismos se basan en un modelo idealizado y no en los hechos observados. Es aquí donde entra en juego la tecnología *Process Mining*. *Process Mining* busca cubrir la brecha entre BPM y BI combinando datos de eventos y modelos de proceso. [3] [2] [9] [10]

Como mencionamos previamente la idea básica de *Process Mining* es extraer conocimiento de logs de eventos obtenidos de distintos sistemas de información, para esto existen un conjunto de técnicas que se agrupan según su funcionalidad en:

- Técnicas de descubrimiento de proceso
- Técnicas de chequeo de concordancia del proceso con la realidad
- Técnicas de extensión y mejora del proceso

A continuación propondremos un enfoque que utiliza estos distintos tipos de técnicas de *Process Mining* como medio para analizar los procesos que se encuentran implantados en un BPMS. Para comprender este enfoque comenzaremos analizando el ciclo de vida de procesos en BPM y como se puede vincular con *Process Mining* y sus técnicas.

3.1 Análisis ciclo de vida de procesos en BPM y vinculación con *Process Mining*

El ciclo de vida de procesos en BPM está compuesto por cuatro etapas principales:

- Diagnóstico y obtención de requerimientos: en esta etapa se hace un relevamiento de los requerimientos y de todos los factores que influyen en el proceso para tenerlos en cuenta en la siguiente fase.
- (re) Diseño: En esta fase se elabora un modelo representativo de los requerimientos. Aquí entra en juego la interpretación de los mismos por los analistas de procesos, que buscan representar la realidad de la manera más representativa.
- Implementación: Este modelo se transforma en un sistema ejecutable en la etapa de configuración/implementación.
- Ejecución/Monitoreo: Luego comienza la fase de monitoreo, en esta etapa los procesos se ejecutan y mientras se monitorean por administradores para ver si algún cambio es necesario.

Algunos de los cambios son tomados en la etapa de **ajuste**. En esta etapa no se crea nuevo software ni se rediseña el existente, solo se utilizan controles predefinidos para adaptar o reconfigurar el proceso.

En la etapa de diagnóstico y requerimientos se evalúa el proceso y se monitorea requerimientos emergentes debido a cambios en el entorno del proceso.

Una pobre performance o la imposición de nuevas demandas del medio ambiente pueden generar una nueva iteración en el ciclo de vida de BPM, comenzando en la fase de rediseño.

La Figura 18 ilustra el ciclo de vida de procesos de BPM descripto.

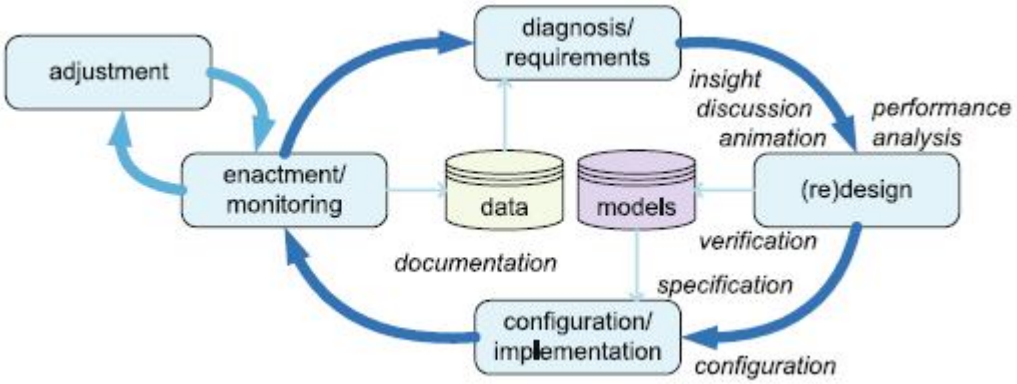


Figura 18 Ciclo de vida de procesos en BPM influencia de los modelos y los datos en cada etapa [4]

Como puede verse en la **Figura 18** el modelo de proceso juega un papel fundamental en las etapas de (re)diseño y configuración/implementación, mientras que los datos tienen un rol dominante en las etapas de monitoreo y diagnóstico y obtención de requerimientos.

Hasta hace un tiempo atrás, no había conexión entre los datos producidos por la ejecución de los procesos y el proceso de diseño actual. Además, el ciclo de BPM solo se reiniciaba cuando había un cambio externo muy importante.

Es aquí donde entra en juego *Process Mining* que aplica sus técnicas basándose en estos dos pilares: datos y modelos. Y que brinda distintos tipos de técnicas que se pueden utilizar como soporte en cada una de las etapas del ciclo de vida de BPM:

Para la **etapa de diagnóstico y obtención de requerimientos** *Process Mining* brinda técnicas que permiten hacer un sondeo sobre el proceso y obtener información general del mismo tal como: el *log inspector* y el *dotted chart.*

Para la **etapa de diseño** brinda técnicas con distintos grados de complejidad de descubrimiento de patrones que permiten descubrir un modelo de proceso.

Para la **etapa de configuración/ implementación** *Process Mining* posee técnicas que permiten extender el modelo y agregar las perspectivas organizacionales, de tiempo y de caso.

Para la **etapa de ejecución y monitoreo** brinda distintos tipos de técnicas como diagramas de transiciones, modelos *fuzzy* y técnicas de análisis BPM que permiten monitorear el proceso, hacer predicciones, analizar la performance, detectar cuellos de botella, entre otras tareas. [2][3][8]

En la implementación del ejemplo práctico se ve el uso de cada una de las técnicas mencionadas, permitiendo ver la potencialidad de las mismas.

3.2 Enfoque para aplicación de *Process Mining* sobre un BPMS

El enfoque propuesto por *Process Mining* para obtener un modelo completamente integrado cubriendo todos los aspectos relevantes del proceso (**Figura 19)**, es aplicable a cualquier organización donde no se tenga siquiera un proceso, o en alguna donde se tenga uno y el mismo se desea extender y mejorar.

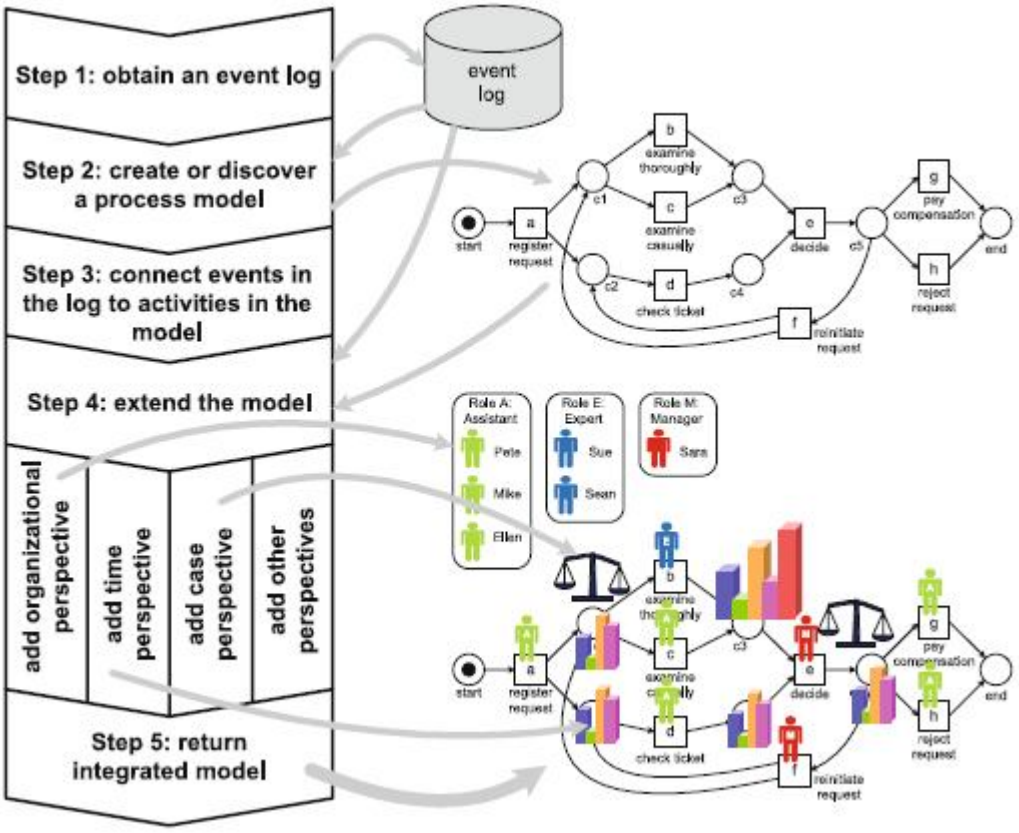


Figura 19 Enfoque para obtener un modelo completamente integrado cubriendo la perspectiva de tiempo, organizacional y de caso.

El enfoque consiste de cinco pasos:

- **Paso 1:** obtener el log de eventos. Aquí es cuando se extraen los datos que conformarán el log de una variedad de sistemas de información, esta acción debe ser llevada a cabo mediante una herramienta generadora de log de eventos. En este paso es fundamental tener un panorama completo del proceso y un conocimiento de la ubicación de los datos.
- **Paso2:** crear o descubrir un modelo de proceso, este paso puede realizarse utilizando alguna de las técnicas de descubrimiento de procesos provistas por *Process Mining* o bien ser creado convencionalmente como por ejemplo por los analistas del proceso.
- **Paso 3:** conectar eventos en el *log* a actividades en el modelo: este paso es esencial para proyectar información en los modelos y agregar perspectivas. Utilizando la técnica de *replay* se conectan los eventos y las actividades en el modelo.
- **Paso 4:** extender el modelo:

- **Agregar la perspectiva organizacional:** es posible analizar la red de trabajo social y subsecuentemente identificar entidades organizacionales que conectan actividades en grupos de recursos
- **Agregar la perspectiva tiempo:** los *timestamps* y frecuencias se pueden usar para aprender distribuciones de probabilidad que describen adecuadamente los tiempos de espera y tiempos de servicios y probabilidad de ruteo.
- **Agregar la perspectiva de caso:** por medio de Minería de Decisión se pueden usar los atributos en el log. Esto muestra como los datos son relevantes y deben ser incluidos en el modelo.
- **Agregar otra perspectiva:** dependiendo de la información en el log otras perspectivas se pueden agregar al modelo. Por ejemplo, información y riesgos y costos se pueden agregar al modelo.

- **Paso5:** devolver el modelo integrado, listo para ser analizado con las distintas técnicas de *Process Mining*.

El modelo integrado resultante provee una vista holística del proceso. Esto provee nuevos puntos de vista y puede generar varias ideas para el mejoramiento del proceso. [2] [3]

Si quisiéramos implementar un proceso en un BPMS desde el principio, obteniendo el log de eventos de los distintos sistemas de información en los que el proceso puede haber dejado un "rastro" y mediante la aplicación del enfoque propuesto de *Process Mining*, se obtendrá un panorama completo del proceso resultando *Process Mining* una herramienta de soporte y ayuda para la implementación del proceso ayudándonos a tener una visión más clara de los requerimientos y de la realidad del proceso, asistiéndonos en el modelado, detectando roles y ayudándonos en la implementación y monitoreo del mismo.

Sin embargo, si quisiéramos aplicar el enfoque a un proceso ya implantado en un BPMS e utilizar las técnicas de *Process Mining* para la optimización del mismo, el enfoque sería distinto por varias razones:

- Ya se dispone de un modelo de proceso.
- Como el proceso ya se encuentra en producción, el mismo genera un historial en la base de datos del BPMS mismo, y este estaría compuesto por la instancia de proceso y las variables de la misma.
- La conexión de los eventos con las actividades del modelo se realiza innatamente por el BPMS, la base de datos del mismo almacena las instancias de procesos ejecutadas registrando para cada evento a que actividad del modelo corresponde.
- Un BPMS ya contempla las cuatro perspectivas: La perspectiva organizacional ya se encuentra relacionada al modelo porque cada actividad o *lane* debe tener un actor. En cuanto a la perspectiva de tiempo un BPMS siempre registra con *timestamps* la fecha de ocurrencia de todos los eventos. La perspectiva de caso está cubierta por las variables de proceso y de actividad que difieren de un caso a otro.

Se puede concluir entonces que un proceso implementado en un BPMS es un buen escenario para la aplicación de *Process Mining* ya que el procesamiento que se debe realizar para aplicar las técnicas se disminuye.

En definitiva, la problemática se reduce a construir el log de eventos en el formato indicado para poder ser importado en una herramienta de *Process Mining* y aplicarle las distintas técnicas de optimización, chequeo de concordancia y performance que nos brinda *Process Mining*.

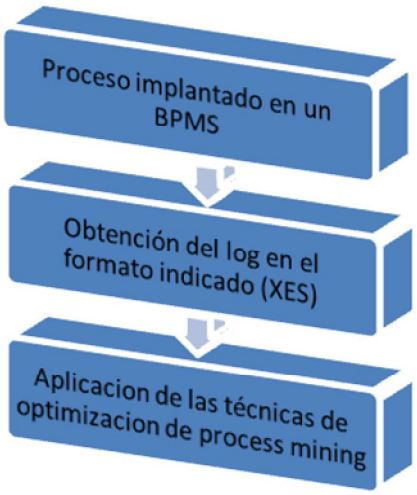


Figura 20 Procedimiento para aplicar las técnicas de *Process Mining* a un proceso implantado en un BPMS.

Como se muestra en la **Figura 20** el procedimiento para aplicar las técnicas de optimización de *Process Mining* a un proceso implantado en un BPMS es el siguiente:

- En una primera etapa se implementó el proceso en un BPMS con o sin asistencia de las técnicas de *Process Mining*, y ahora se quiere utilizar la misma para medir la performance, monitorear y optimizar el proceso.
- En la segunda etapa se debe tener un conocimiento de la ubicación de los datos en la base de datos del BPMS. Luego mediante una herramienta de generación de logs de eventos en formato XES extraen los datos y se genera el log.
- Una vez generado el log mediante una herramienta de *Process Mining* se puede importar el mismo y se le pueden aplicar las técnicas de optimización y monitoreo.

Como conclusión nos podemos encontrar con dos escenarios para la aplicación de *Process Mining* sobre un BPMS:

- Primer escenario: No se encuentra modelado el proceso para la cual se recomienda aplicar el primer enfoque mencionado de manera de obtener una vista general del proceso y luego implementar el proceso en el BPMS.
- Segundo escenario: el modelo ya se encuentra implementado y corriendo y se desea utilizar las técnicas como método de optimización, para chequear la credibilidad del proceso, analizar la performance del mismo, y monitorearlo.[1][2][3][8]

Luego de haber visto el enfoque propuesto para la aplicación de las técnicas de *Process Mining* sobre un proceso implantado en un BPMS, nos encontramos en

condiciones de focalizarnos en la segunda etapa del enfoque **Obtención del log en el formato XES,** aquí automatizaremos esta tarea, tomando como herramientas de desarrollo Bonita Open Solution y ProM. En la siguiente sección se justificará la elección de estas herramientas y se entrará en detalle en la implementación de la integración.

Capítulo 4: Mecanismo de integración entre Bonita Open Solution y ProM

En este capítulo se propone un mecanismo de integración entre Bonita Open Solution y ProM, que sistematice la construcción del log de eventos extendiendo la funcionalidad del Bonita Open Solution, luego el log de eventos puede ser tomado como entrada en la herramienta de *Process Mining*, ProM.

Primero se verán las herramientas a utilizar para el desarrollo, y el motivo de selección de cada una de ellas. Luego se realizara un análisis sobre las ventajas de sistematizar la construcción del log de eventos, haciendo hincapié en la implementación del conector. Finalmente se enunciaran las buenas prácticas para la implementación del generador del log de eventos en cualquier BPMS.

4.1 ProM

ProM es un framework open source genérico para implementar técnicas de *Process Mining* en un ambiente estándar. El framework ProM recibe como entrada logs de eventos en formato XES o MXML. Actualmente el framework tiene plug-ins para *Process Mining*, análisis, monitoreo y conversión.

ProM está disponible en archivos de distribución binaria para plataformas Windows, Mac, y Unix. La última versión de ProM tiene licencia GPL.

Los productos de BI se centran en los datos y son limitados cuando se requiere formas más avanzadas de análisis. Las herramientas de data mining ofrecen más "inteligencia", pero también se centran en los datos.
Los sistemas que se centran en el proceso como sistemas BPM, herramientas de simulación, y herramientas de modelado, se focalizan más en Play-out (ver figura Play out). Las plataformas de software tradicional no permitían la utilización de técnicas de *Process Mining*. Esto disparó el desarrollo de varias herramientas de *Process Mining* que se dedicaban a una técnica especial de *Process Mining*. Claramente no tenía sentido dedicar una herramienta de *Process Mining* para cada técnica de descubrimiento de proceso nueva. Esta observación provocó el desarrollo de Prom framework, un entorno "plug-able" para *Process Mining* que utilizaba MXML como formato de entrada.

A lo largo de las distintas versiones de ProM, más plugins para soportar distintas técnicas de data mining fueron incorporados. De esta forma ProM se convirtió el estándar de facto para *Process Mining*. Grupos de investigación de todo el mundo contribuyen en el desarrollo de ProM y cientos de organizaciones han descargado ProM.

La última versión de ProM (ProM 6) está basada en XES más que en MXML. XES es el nuevo estándar de *Process Mining* adoptado por la IEEE Task Force for *Process Mining*.

Para generar los logs en formato XES, ProM cuenta con una aplicación llamada XESame que permite la extracción de un log de evento de fuentes de datos.

El principal motivo de selección de la herramienta ProM es su carácter open source, gratuito, sumado a la cantidad de documentación disponible gracias a los distintos aportes realizados por distintos grupos de investigadores de todo el mundo. [4][6]

4.2 Bonita Open Solution

Bonita Open Solution [8] (Solución Abierta Bonita) es una suite ofimática para la Gestión de procesos de negocio y realización de Workflows, creada en 2001. Es código abierto y puede ser descargado bajo GPL v2.

Características generales:

- Está compuesto por dos ambientes, producción y desarrollo, este último por 3 elementos: Bonita Studio, BonitaBPM engine, BonitaUserXP
    - Los usuarios utilizan Bonita Studio para diseñar procesos de manera natural e intuitiva con BPMN
    - En Bonita Studio los usuarios pueden integrar funcionalidades de sistemas externos por medio de los más de 100 conectores existentes, o crearlos ellos mismos
    - En Bonita Users XP la ejecución de los procesos puede ser emulada por los distintos participantes de los procesos, accediendo al portal web que maneja las tareas a modo de e-mal inbox.
    - El Bonita Engine puede ser extendido para integrar nuevos servicios, es suficientemente flexible y potente
- El deploy de los procesos simplemente consta de la importación de los mismos en el ambiente de producción, el cual es igual al Bonita User XP

Tabla 3 Aspectos Generales Bonita Open Solution

	Instalación	Licenciamiento	Motor de reglas	Facilidad de integración
Bonita	Instalación rápida, requerimientos de plataforma Java, SDK, tiempo estimado: 5 minutos	Open source en la versión investigada. Existe versión SP que incluye funcionalidades como la administración de privilegios	No posee motor de reglas, sin embargo implementa tablas de decisión. Permite conectar con un motor externo como Drools	Permite la integración de otros sistemas y servicios web a través de conectores así como embeber formularios en aplicaciones web.

A continuación realizaremos un análisis de las funcionalidades provistas para todo el ciclo de vida de BPM, es decir, para la etapa de modelado, implementación, ejecución y monitoreo.

En la siguiente tabla se puede observar a grandes rasgos la performance de bonita con respecto al ciclo de vida de BPM en general, luego se verán en detalle cada una de las etapas.

Tabla 4 performance de Bonita Open Solution del ciclo de vida en general

	Etapa	Bonita
Ciclo de vida	Modelado	Completo y rico en elementos
	Implementación	Completa
	Ejecución	Buena
	Gestión y monitoreo	Básica

Como se puede ver Bonita presenta un modelador de procesos completo y ricos en elementos, la implementación es completa, la ejecución de procesos es buena mientras que la gestión y el monitoreo de procesos es básico, no permite BAM, en la versión gratuita y el monitoreo de procesos es prácticamente nulo.

A continuación analizaremos en detalle cada una de las etapas:

Tabla 5 características de Bonita Open Solution en cuanto a la etapa de modelado

	Facilidades para modelar	Cumplimiento del estándar BPMN	Observaciones extra	Inicio de los procesos
Bonita	Intuitivo y simple, permite hacer una representación fiel de un proceso a quien lo modele. Completo en elementos para el diseño, varios tipos de tarea.	Se apega al estándar, define pools y lanes, distintos tipos de tareas y compuertas.	Permite que las tareas puedan autoejecutarse, si se configuran como tarea de servicio. Se puede asignar participantes a un Lane para que todas las tareas pertenecientes al Lane sean se ejecuten con por el mismo.	Ejecución desde la User XP. Inicio disparado por un mensaje entre procesos. Invocando desde la API de Java.

Tabla 6 Características de Bonita Open Solution en cuanto a la etapa de implementación

	Editor de formularios	SOA	Interacción con base de datos	Variables de proceso
Bonita	Bueno, permite definir el orden de los campos y su tipo, a su vez se pueden insertar scripts o atributos HTML así como la inserción de formularios en aplicaciones web, fácil de aprender a utilizar.	Permite el consumo de un servicio web por medio de su principal puerta de interconexión que son los conectores. Gracias a la arquitectura basada en Java, es fácil crear servicios web que se comuniquen con los procesos.	MySQL, Postgre SQL, Oracle DB y la posibilidad de agregar otros conectores para ejecutar consultas H2 y otros gestores de BD.	Las variables del proceso se pueden definir de manera global al proceso o de manera local a la tarea, se definen características como nombre, tipo y valor predeterminado.

Tabla 7 Características de Bonita Open Solution en cuanto a la etapa de ejecución

	Ambientes e Intervención de los procesos en tiempo real	Depurador
Bonita	Permite intervención de los procesos en ambiente de desarrollo, el cual provee la funcionalidad de emular el proceso tal cual funcionaria en el ambiente de producción, luego este ambiente importa el proceso desarrollado.	No posee depurador, se puede acceder al log del motor, pero no es la opción más amigable.

Tabla 8 Características de Bonita Open Solution en cuanto a la etapa de gestión y monitoreo

	Reportes y dashboards (BAM)	Simulación de procesos
Bonita	Permite generar dashboards y reportes en base a índices predeterminados, como tiempos promedio de ejecución o número de casos comenzados, no permite definir índices específicos..	Simple, permite definir valores, condiciones y recursos para evaluar situaciones particulares generando reportes.

Como puede verse Bonita Open Solution es un BPMS completo, fácil de utilizar y modificar, gracias a su carácter de herramienta open source. [8]

4.2.1 Arquitectura Bonita Open Solution

En la Figura 21 se puede ver como es la arquitectura run-time de bonitaSoft, esta se compone del Bonita User Experience "User XP" y de Bonita Execution Engine. Bonita User XP es la consola de usuario donde los mismos se loguean y pueden administrar las tareas y monitorear los procesos. Bonita Execution Engine es el motor de ejecución de Bonita que está compuesto por el contenedor de servicios, la API y los conectores. El contenedor de servicios provee servicios de identidad, de almacenamiento persistente, transacción, entre otros. La API de Bonita está compuesta por la API de BAM, la API de administración de usuarios, la API de administración de tareas y la API de administración de procesos.

El módulo de Bonita UserXP se conecta al Execution Engine por medio de la API. Las aplicaciones de Bonita y las aplicaciones propias de usuario también se conectan al Execution Engine a través de la API.

Por medio de los conectores Bonita se comunica con otros sistemas externos, estos son la puerta de Bonita hacia el exterior. [8]

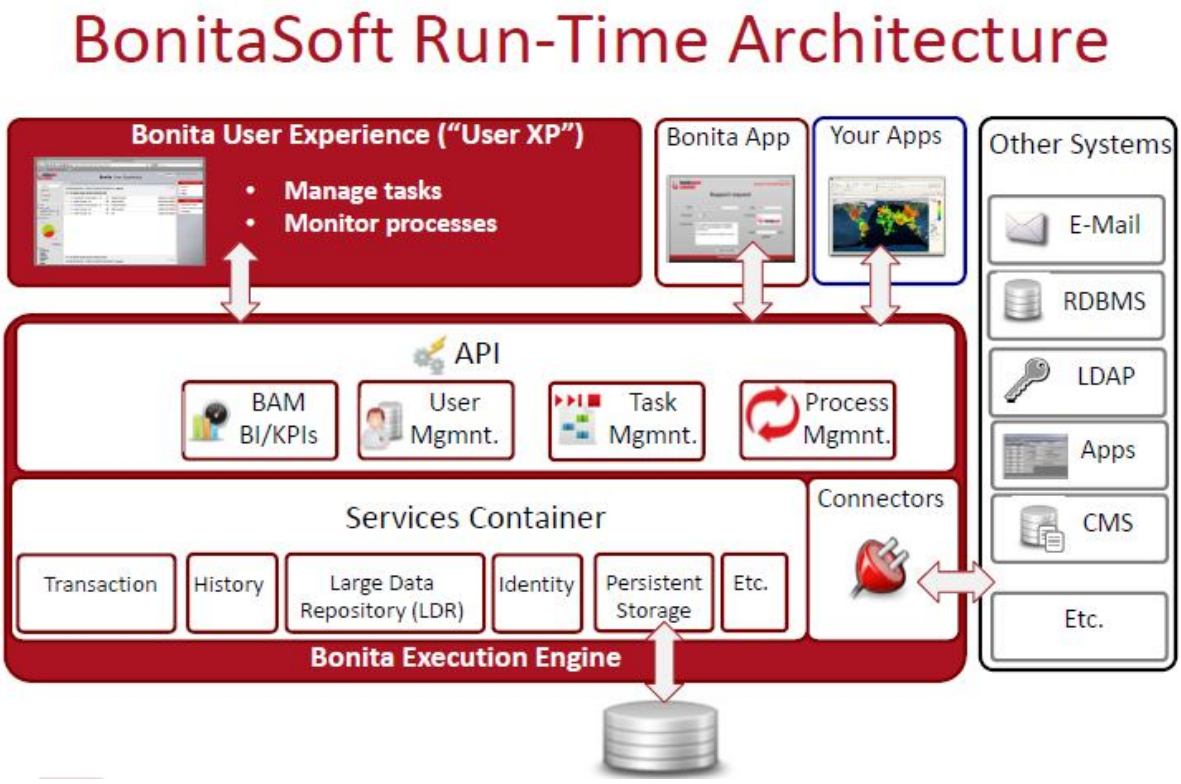


Figura 21 Arquitectura Run-Time BonitaSoft [8]

4.2.2 Capacidad de Bonita Open Solution de integración con aplicaciones externas mediante conectores

Bonita Open Solution provee "conectores" para conectar tareas (actividades) o procesos (pool) a sistemas de información externos.

Los conectores toman una entrada específica (directamente como un valor del usuario final o reconstituido o incorporado en una expresión) y ejecutan código Java. Algunos conectores también devuelven una salida a Bonita Open Solution.

Hay dos maneras de configurar conectores:

- Agregar un conector: elegir un conector que ya se encuentre definido en Bonita Open Solution
- Crear un conector: crea un nuevo tipo de conector, ubicarlo en una categoría de Bonita Open Solution existente, utilizarlo como wizard para futuros conectores del mismo tipo. Una vez que el conector se crea, se puede mover o copiar (a otra tarea o proceso /pool).

Los conectores de Bonita Open Solution aceptan código embebido, toman una entrada, se comunican con servicios externos, y retornan salidas como se ve en la Figura 64.

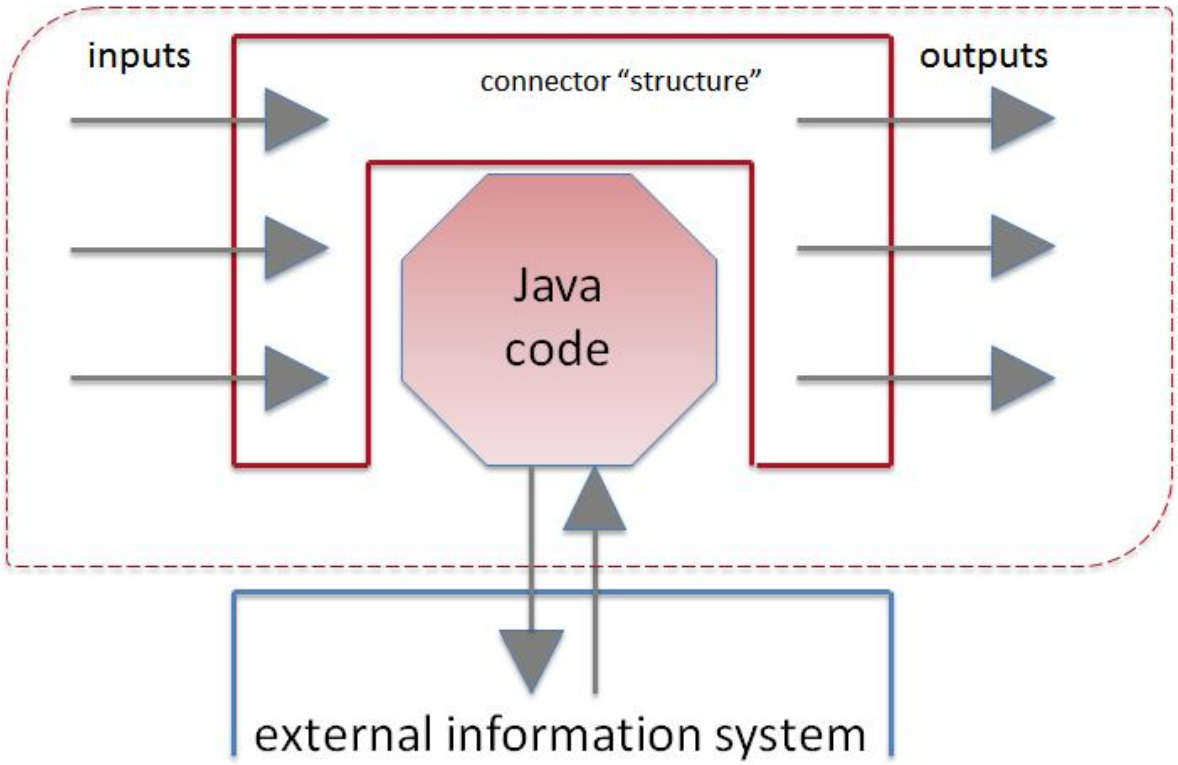


Figura 22 Los conectores de Bonita Open Solution aceptan código embebido.[8]

Se ha elegido Bonita Open Solution como el BPMS para el desarrollo de este trabajo por ser unas herramientas open source, fácil de utilizar, y con una amplia capacidad de extensión gracias a los conectores que no solo sirven para comunicarse con sistemas externos sino para agregar funcionalidad al BPMS como mostraremos en la siguiente sección. La versión de Bonita elegida para realizar el desarrollo es la 5.8. [8]

4.3 Proceso generador del log de eventos

Como se mencionó en el capítulo 1, las técnicas de *Process Mining* permiten extraer conocimiento de un log de eventos. Hoy en día la información para crear un log de eventos se encuentra disponible en la mayoría de los sistemas de información. Estas técnicas proveen nuevas formas de descubrir, monitorear y mejorar procesos en una variedad de dominios de aplicación.

El log de eventos para poder ser utilizado como entrada en una herramienta de *Process Mining* debe cumplir con ciertas características y formato:

- Debe estar escrito bajo el estándar XES
- Un log de eventos es por una única definición de proceso
- Los eventos deben ser almacenados secuencialmente por orden de ocurrencia
- Cada evento **referencia** a una actividad (event)
- Cada evento se **relaciona** a un caso particular (trace)

El log de evento se utiliza para aplicar 3 tipos distintos de técnicas de *Process Mining*:

- Descubrimiento: produce un modelo a partir de un log
- Chequeo de concordancia: chequea si la realidad almacenada en el log se adecua al modelo
- Mejoramiento: dado un log de eventos y un modelo mejorar y extender el modelo

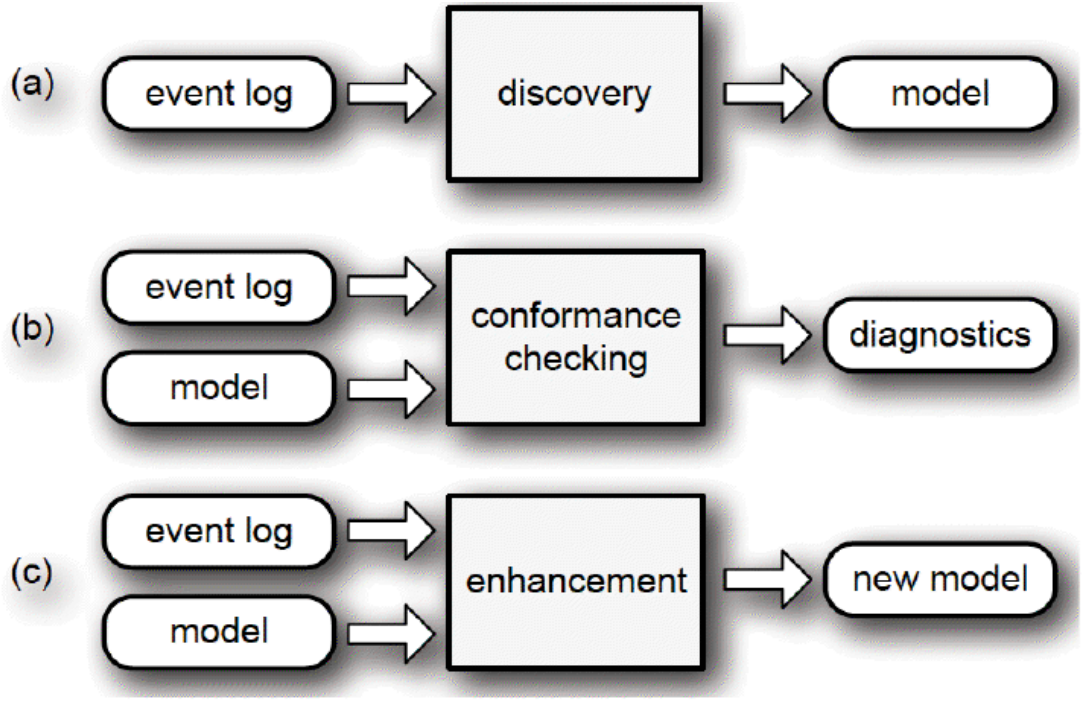


Figura 23 Rol del log de eventos para aplicar las técnicas de *Process Mining* [4]

Como se ve en la figura 23 las técnicas de descubrimiento toman como entrada el log de eventos, y como salida devuelven un modelo de proceso (en una red de Petri, en notación BPM) o también puede entregar modelos de otras perspectivas como por ejemplo una red de trabajo social.

Las técnicas de chequeo de concordancia toman como entrada un log de eventos y un modelo de proceso existente, y como salda producen información de diagnóstico mostrando discrepancias entre el modelo y el log.

Re pasemos las perspectivas en las que se focaliza *Process Mining* para analizar el log de eventos:

- **Perspectiva de flujo de trabajo**: se centra en el orden de las actividades. El objetivo es encontrar una buena caracterización de los caminos posibles.
- **Perspectiva organizacional:** se centra en la información de los recursos que se encuentran involucrados y como se relacionan.
- **Perspectiva de tiempo:** se refiere a la temporización y a la frecuencia de eventos, ayuda a analizar la performance del proceso.
- **Perspectiva de caso:** se focaliza en propiedades específicas del caso (monto de una compra, nacionalidad de un cliente).

En la siguiente tabla se mostraran los pasos que se deben seguir para aplicar las técnicas de *Process Mining* sobre un log generado por un proceso implantado en cualquier BPMS, sin la implementación del proceso generador de log de eventos:

Tabla 9 Pasos para la aplicación de *Process Mining* sobre un proceso implantado en un BPMS sin proceso generador de log

Paso	Requiere
1. Extraer log de eventos con XESame	Se debe tener un conocimiento de la ubicación de los campos de interés en las tablas de la base de datos del BPMS. Se deben setear las extensiones del log, seleccionar atributos globales de trace y de evento; y setear clasificadores. XESame no es una herramienta intuitiva por lo que se requiere de un esfuerzo adicional para la configuración del mapeo.
2. Cada vez que se desea aplicar una técnica de *Process Mining* se debe aplicar sobre el log actualizado	Volver a correr el mapeo generado en XESame
3. Cargar log de eventos en ProM y aplicar técnicas de *Process Mining*	Conocimiento de la conformación del log de eventos, y de las técnicas de *Process Mining* aplicables al mismo

Como puede verse la utilización de la herramienta XESame para obtener el log de eventos del proceso implantado en un BPMS es un paso que requiere, esfuerzo y tiempo por parte de los analistas, que deben hacerse interiorizarse en los detalles de implementación del BPMS particular.

En el caso de Bonita, por defecto la base de datos es H2 y se encuentra oculta, por lo tanto debería hacerse una instalación personalizada donde se instale bonita en una base de datos accesible como por ejemplo Mysql. Bonita cuenta con dos bases de datos: bonita_journal (en esta base de datos se encuentran las instancias de procesos que no han culminado) y bonita-history (en esta base de datos se encuentran las instancias de proceso ya finalizadas). Cada una de estas bases de datos cuenta con 68 tablas. El estudio de la ubicación de los datos en la base de datos de bonita y como realizar los JOINS correspondientes para traer la información por cada trace es tarea de los analistas y requiere de mucho tiempo de investigación.

Con la implementación del proceso generador del log de eventos se sistematiza la construcción del log de eventos facilitando la tarea de los analistas ya que independiza de la base de datos y accede a los datos de proceso por medio de la API.

4.3.1 Implementación del proceso generador del log de eventos

Para la implementación del proceso se utilizó la versión 5.8 de Bonita Open Solution. El diagrama de proceso se muestra en la Figura 24.

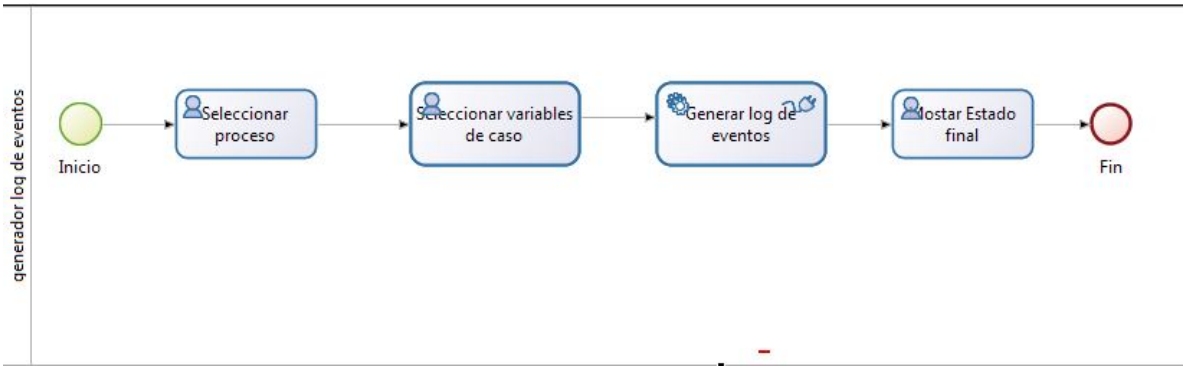


Figura 24 Diagrama del proceso generador del log de eventos

El proceso está compuesto por 4 actividades secuenciales "Seleccionar proceso", "Seleccionar variables de caso", "Generar log de eventos" y "Mostrar Estado final". Las dos primeras son actividades manuales, la tercera es una tarea automática que contiene al conector generador del log de eventos y la ultima simplemente muestra el resultado final del proceso.

En la tarea "Seleccionar proceso" se despliega un formulario donde se despliega un listbox con todas las definiciones de proceso disponibles en el motor de procesos. Se debe seleccionar el proceso del que se desea generar el log de eventos, y luego presionar el botón continuar como se muestra en la figura 25.

Figura 25 Formulario de la tarea "Seleccionar proceso"

El siguiente código fue utilizado como entrada en el widget listbox, mediante la API de definiciones de Bonita, se obtienen los procesos y se despliegan en la lista los nombres de los mismos.

```
     final QueryDefinitionAPI queryDefinitionAPI =
AccessorUtil.getQueryDefinitionAPI();
     List procesos= (ArrayList)
queryDefinitionAPI.getProcesses();

     List nombres = new ArrayList();
     for (  LightProcessDefinition p : procesos){
          nombres.add(p.getUUID());
     }
     return nombres;
```

En la tarea "Seleccionar variables de caso" se despliega un formulario que muestra en un checkboxlist las variables de proceso disponibles, y el usuario debe seleccionar cuales desea almacenar en el log de eventos. También se debe indicar la ruta de almacenamiento donde se guardara el log de eventos, y el nombre que tendrá el log. Un ejemplo se muestra en la figura 26.

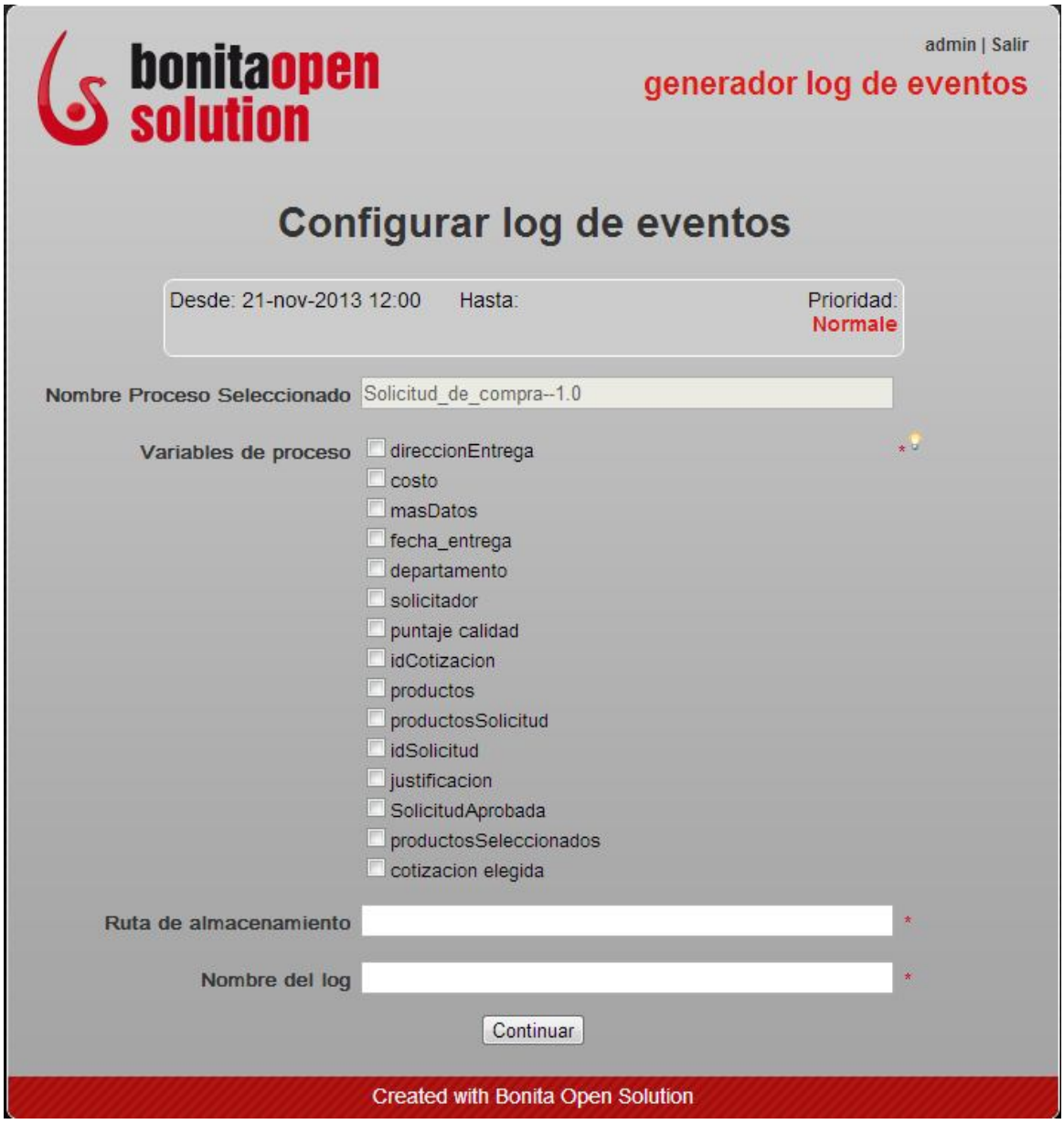


Figura 26 Formulario de la tarea "Seleccionar variables de caso"

El siguiente código fue utilizado como entrada en el widget variables de proceso. Mediante la API de definiciones de Bonita, se obtienen los nombres de las variables de la definición de proceso seleccionada en el formulario anterior

```
     final QueryDefinitionAPI queryDAPI =
AccessorUtil.getQueryDefinitionAPI();
     ProcessDefinitionUUID pd= new
ProcessDefinitionUUID(nombreProceso);
     List variables= (ArrayList)
queryDAPI.getProcessDataFields(pd);

     List nombres = new ArrayList();
```

```
for (  DataFieldDefinition dfd : variables){
     nombres.add(dfd.getLabel());
}
return nombres;
```

La tercer tarea "Generar log de eventos" es una tarea automática que contiene únicamente un conector que se encarga de generar el log de eventos con las variables seteadas en las tareas previas.

El conector insertado en esta tarea es un conector de tipo Script, en particular un conector de scrpit groovy que es código Java. Este conector es un código que se ejecuta al principio o al inicio de la tarea y mediante el cual se permite agregar funcionalidad a la herramienta.

En la implementación el conector se configuro para que se ejecute al inicio de la tarea automática.

A continuación detallaremos cada uno de los métodos del Script Groovy.

El script principal invoca un método que crea el documento XML, luego invoca a un método dado un documento XML agrega el elemento raíz y lo devuelve, luego invoca a un método que agrega los atributos del caso y finalmente invoca a un método que escribe el documento XML.

```
try {
     Document doc= crearXML();
     Element log= crearLog(doc);
     AgregarAtributosDeTrace(doc, log);
     EscribirDocumento(doc);
     System.out.println("File Saved!");
} catch (IOException io) {
     System.out.println(io.getMessage());
}
```

El siguiente es el código del método "crearXML" que crea al documento XML, para ello utiliza DocumentBuilderFactory.

```
//metodo que crea el documento XML
private Document crearXML(){
     try {
          DocumentBuilderFactory dbfac =
DocumentBuilderFactory.newInstance();
          DocumentBuilder docBuilder;
          docBuilder = dbfac.newDocumentBuilder();
          Document doc = docBuilder.newDocument();
          doc.setXmlVersion("1.0");
          doc.setXmlStandalone(false);
          return doc;
     } catch (IOException io) {
     System.out.println(io.getMessage());
}
}
```

El siguiente es el código del método “crearLog” que crea el elemento log en el documento XES, con todas las extensiones: ciclo de vida, organizacional, de tiempo, de concepto y semántica que son las extensiones del estándar XES, como se vio en el anexo 2. Se agregan los atributos globales del trace y del evento. Se agregan los clasificadores y el estado y nombre del proceso perteneciente al log. Finalmente se devuelve el elemento log con todos sus atributos y elementos configurados.

```
     private Element crearLog(Document doc){
          Element log = doc.createElementNS("http://www.xes-
standard.org/", "log");
          log.setAttribute("xmlns", "http://www.xes-
standard.org/");
          log.setAttribute("openxes.version", "1.0RC7");
          log.setAttribute("xes.features", "nested-
attributes");
          log.setAttribute("xes.version", "1.0");
          doc.appendChild(log);
          //se agregan extensiones

          Element extension = doc.createElement("extension");
          extension.setAttribute("name", "Lifecycle");
          extension.setAttribute("prefix", "lifecycle");
          extension.setAttribute("uri", "http://www.xes-
standard.org/lifecycle.xesext");
          log.appendChild(extension);
          extension = doc.createElement("extension");
          extension.setAttribute("name", "Organizational");
          extension.setAttribute("prefix", "org");
          extension.setAttribute("uri", "http://www.xes-
standard.org/org.xesext");
          log.appendChild(extension);
          extension = doc.createElement("extension");
          extension.setAttribute("name", "Time");
          extension.setAttribute("prefix", "time");
          extension.setAttribute("uri", "http://www.xes-
standard.org/time.xesext");
          log.appendChild(extension);
          extension = doc.createElement("extension");
          extension.setAttribute("name", "Concept");
          extension.setAttribute("prefix", "concept");
          extension.setAttribute("uri", "http://www.xes-
standard.org/concept.xesext");
          log.appendChild(extension);
          extension = doc.createElement("extension");
```

```
            extension.setAttribute("name", "Semantic");
            extension.setAttribute("prefix", "semantic");
            extension.setAttribute("uri", "http://www.xes-
standard.org/semantic.xesext");
            log.appendChild(extension);

            //se agregan atributos globales del trace
            Element global= doc.createElement("global");
            global.setAttribute("scope", "trace");
            Element atglobal= doc.createElement("string");
            atglobal.setAttribute("key", "concept:name");
            atglobal.setAttribute("value", "UNKNOWN");
            global.appendChild(atglobal);
            atglobal= doc.createElement("string");
            atglobal.setAttribute("key",
"semantic:modelReference");
            atglobal.setAttribute("value", "UNKNOWN");
            global.appendChild(atglobal);
            log.appendChild(global);

            //se agregan atributos globales de evento
            global= doc.createElement("global");
            global.setAttribute("scope", "event");
            atglobal= doc.createElement("string");
            atglobal.setAttribute("key", "concept:instance");
            atglobal.setAttribute("value", "UNKNOWN");
            global.appendChild(atglobal);
            atglobal= doc.createElement("string");
            atglobal.setAttribute("key", "org:resource");
            atglobal.setAttribute("value", "UNKNOWN");
            global.appendChild(atglobal);
            atglobal= doc.createElement("date");
            atglobal.setAttribute("key", "time:timestamp");
            atglobal.setAttribute("value", "1970-01-
01T00:00:00-03:00");
            global.appendChild(atglobal);
            atglobal= doc.createElement("string");
            atglobal.setAttribute("key",
"lifecycle:transition");
            atglobal.setAttribute("value", "UNKNOWN");
            global.appendChild(atglobal);
            atglobal= doc.createElement("string");
            atglobal.setAttribute("key", "concept:name");
            atglobal.setAttribute("value", "UNKNOWN");
            global.appendChild(atglobal);
            log.appendChild(global);
```

```
            //se agregan los clasificadores
            Element classifier=
doc.createElement("classifier");
            classifier.setAttribute("keys", "concept:name
lifecycle:transition");
            classifier.setAttribute("name", "Activity
classifier");
            log.appendChild(classifier);
            classifier= doc.createElement("classifier");
            classifier.setAttribute("keys", "org:resource");
            classifier.setAttribute("name", "Resource
classifier");
            log.appendChild(classifier);

            //se agrega estado y nombre del proceso
perteneciente al log
            Element atributo= doc.createElement("string");
            atributo.setAttribute("key", "concept:name");
            atributo.setAttribute("value", nombreProceso);
            log.appendChild(atributo);
            atributo= doc.createElement("string");
            atributo.setAttribute("key", "lifecycle:model");
            atributo.setAttribute("value", "complete");
            log.appendChild(atributo);

            return log;
      }
```

El siguiente es el código del método AgregarAtributosDeTrace que agrega las instancias de proceso y los valores de las variables seleccionadas para almacenar en el log de eventos. Para poder acceder a los valores de las variables en una determinada instancia de proceso se utiliza la API de ejecución de Bonita que permite acceder a las instancias de proceso. Una vez que se crea el elemento trace y se agregan los valores de las variables de la instancia particular se procede invocando al método AgregarAtributosEventos que se encarga de agregar los atributos estándar para cada una de las actividades del proceso.

```
      private void AgregarAtributosDeTrace(Document doc,
Element log){
            final QueryRuntimeAPI queryRAPI =
AccessorUtil.getQueryRuntimeAPI();
            ProcessDefinitionUUID pd= new
ProcessDefinitionUUID(nombreProceso);
            List instanciasDeProceso= (ArrayList)
queryRAPI.getProcessInstances(pd);
```

```
            Element trace= null;
            Element string= null;
            Map<String, Object> valores=null;
            for (  ProcessInstance p : instanciasDeProceso){
                  trace= doc.createElement("trace");
                  for(String nombreVariable:
variablesSeleccionadas){
                        string=doc.createElement("string");
                        valores= p.getLastKnownVariableValues();
                        if(valores.containsKey(nombreVariable)){
                              string.setAttribute("key",
nombreVariable);
                              string.setAttribute("value",
valores.get(nombreVariable).toString());
                              trace.appendChild(string)
                        }
                  }
                  AgregarAtributosEventos(doc, trace, p);
                  log.appendChild(trace);
            }
      }
```

El siguiente es el código del método *AgregarAtributosEventos* que primero se encarga de ordenar las actividades del proceso según la fecha en la que fueron terminadas. Luego accede a todas las actividades de la instancia de proceso específica y setea los atributos que requiere *Process Mining* para aplicar sus técnicas en todas las perspectivas: perspectiva de flujo de trabajo, perspectiva organizacional, perspectiva de tiempo, perspectiva de caso.

```
      private void AgregarAtributosEventos(Document doc,
Element trace, ProcessInstance p){
            SortedSet<ActivityInstance> actividades = new
TreeSet<ActivityInstance>(new Comparator<ActivityInstance>(){
                  public int compare(Object o1, Object o2){
                        ActivityInstance act1 =
(ActivityInstance)o1;
                        ActivityInstance act2 =
(ActivityInstance)o2;
                        return
act1.getEndedDate().compareTo(act2.getEndedDate());
                  }
            });
            actividades.addAll(p.getActivities());
            Element event= null;
            for(ActivityInstance instanciaActividad:
actividades){
                  event= doc.createElement("event");
```

```
                string= doc.createElement("string");
                string.setAttribute("key",
"concept:instance");
                string.setAttribute("value",
instanciaActividad.getActivityInstanceId());
                event.appendChild(string);

                string= doc.createElement("string");
                string.setAttribute("key", "concept:name");
                string.setAttribute("value",
instanciaActividad.getActivityName());
                event.appendChild(string);


     if(instanciaActividad.getState().toString().equals("FINI
SHED")){
                     string= doc.createElement("string");
                     string.setAttribute("key",
"lifecycle:transition");
                     string.setAttribute("value","complete" );
                     event.appendChild(string);
                }

                string= doc.createElement("date");
                string.setAttribute("key", "time:timestamp");
                string.setAttribute("value",
ISO8601DateParser.format(instanciaActividad.getEndedDate()));
                event.appendChild(string);

                if(instanciaActividad.isTask()){
                     string= doc.createElement("string");
                     string.setAttribute("key",
"org:resource");
                     string.setAttribute("value",
instanciaActividad.getTask().getEndedBy().toString());
                     event.appendChild(string);
                }
                trace.appendChild(event);
           }
     }
```

El siguiente es el código del metodo “EscribirDocuemento” que se encarga de guardar el documento en la ruta de almacenamiento indicada con el nombre de log de eventos indicado.

```
     private void EscribirDocumento (Document doc){
          try {
               TransformerFactory transformerFactory =
TransformerFactory.newInstance();
               Transformer transformer =
transformerFactory.newTransformer();

     transformer.setOutputProperty("{http://xml.apache.org/xs
lt}indent-amount", "2");

     transformer.setOutputProperty(OutputKeys.INDENT, "yes");
               DOMSource source = new DOMSource(doc);
               StreamResult result = new StreamResult(new
File(ruta_almacenamiento+"\\"+nombreLog+".xes"));
               transformer.transform(source, result);
          } catch (IOException io) {
          System.out.println(io.getMessage());
     }
     }
```

El proceso generador de log de eventos podrá ser ejecutado únicamente por usuarios analistas, el mismo se verá en la bandeja de entrada como un proceso más pero tiene la particularidad de que es un proceso que agrega funcionalidad al BPMS, en este caso sistematizar la generación del log de eventos.

Gracias a los conectores de Bonita se puede agregar código que realice alguna funcionalidad específica sin conectarse con sistemas de información externos, es decir mediante los conectores se puede enriquecer la funcionalidad de Bonita como en este caso generar un log de eventos. Ejemplo Figura 27.

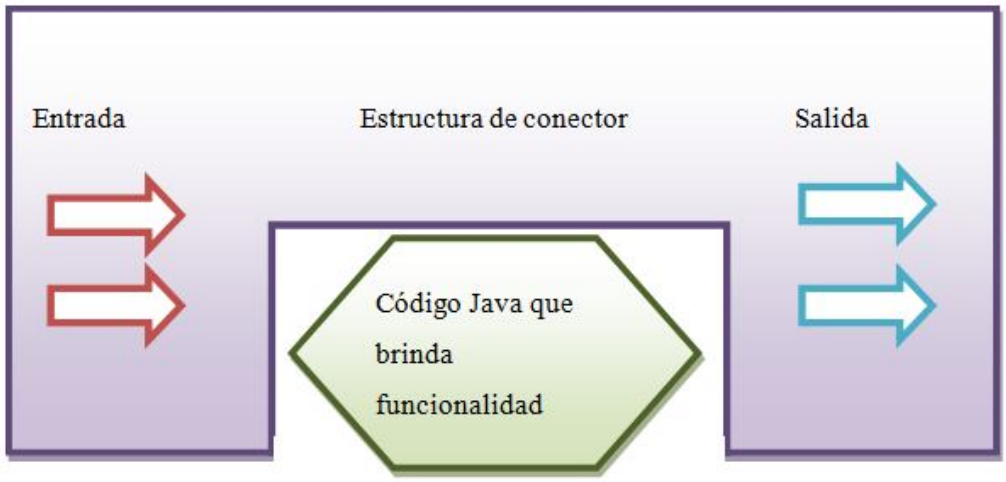


Figura 27 Conector que agrega funcionalidad a Bonita Open Solution

4.3.2 Buenas prácticas para la implementación del proceso generador del log de eventos en cualquier BPMS

Como vimos en el capítulo 2 todos los BPMS tienen una capa de proceso. Lo que la capa de procesos provee, por medio del BPMS, es el nuevo paradigma hacia un framework de integración de aplicaciones. El framework viene con herramientas de desarrollo, conectividad a sistemas comercialmente disponibles, mapeo de datos y otras herramientas. Este framework de integración de aplicaciones permite a las corporaciones crear esquemas de datos para toda la empresa que todas las aplicaciones deben conformar para transferir datos. El framework de integración de aplicaciones representa el eje de datos donde todas las aplicaciones transmiten y reciben información. Todas las aplicaciones se conectan a la capa de proceso por medio de conectores y adaptadores, que son software que encuentran comercialmente disponibles para permitir la conectividad con aplicaciones de negocio. El repositorio de datos en la capa de proceso permite que las aplicaciones obtengan los datos necesarios para sus funciones sin ir directamente a las aplicaciones originarias. Esto elimina la necesidad de interfaces punto a punto y reduce drásticamente el número de conexiones de integración para mantener. [17]

Contemplando este paradigma mediante la implementación de un proceso que extienda la funcionalidad de la herramienta con la ayuda de conectores se pueden generar el log de eventos y eventualmente integrar un BPMS con una herramienta de *Process Mining* ya que por medio del proceso que extiende la funcionalidad se genera automáticamente el log de eventos y este se utiliza como entrada en una herramienta de *Process Mining*.

El log de eventos debe contar con la estructura básica de un documento XES vista en la sección “Logs de eventos, Fuentes de datos”, y debe permitir la configuración de cada uno de los campos del modo más sencillo posible.

El proceso generador de eventos debe contar con dos pasos esenciales:

- Seleccionar el proceso del cual desea generar el log de eventos: mediante la API del BPMS se pueden acceder a los procesos almacenados en el motor de procesos y disponer el listado para la selección del mismo.
- Seleccionar las variables de caso que desean analizarse: mediante la API del BPMS se puede acceder a las variables de caso del proceso seleccionado en el paso anterior.

Una vez configurados estos parámetros, serian la entrada del conector encargado de generar el log de eventos en el formato XES adecuado. Este conector puede estar implementado en el lenguaje de base del BPMS.

Cada BPMS se va a ver limitado por las funcionalidades disponibles en la API y el lenguaje base de la misma.

Capítulo 5: Implementación de un proceso de negocio como caso de estudio

En este capítulo se describirá un proceso de negocio como caso de estudio, y se lo desplegará en la herramienta Bonita Open Solution siguiendo los pasos del ciclo de vida de BPM, vistos en el capítulo 1 (Diagnóstico y obtención de requerimientos, diseño, implementación, ejecución y monitoreo). Luego se utilizará el proceso *generador de log de eventos* descripto en el capítulo anterior y se adicionará al proceso de ejemplo para poder aplicar las técnicas de *Process Mining* con ProM para la etapa de monitoreo y análisis. Finalmente se elaborará un estudio con los beneficios encontrados de la aplicación de las técnicas de *Process Mining* al proceso implementado.

5.1 Diagnóstico y obtención de requerimientos

Un proceso es un conjunto de uno o más procedimientos o actividades directamente ligadas, que colectivamente realizan un objetivo del negocio, normalmente dentro del contexto de una estructura organizacional que define roles funcionales y relaciones entre los mismos. (WFMC. *Workflow Management Coalition*)

Si observamos cualquier empresa de la actualidad, podemos apreciar que la misma está constituida por varias partes (Sectores funcionales). Son estas partes quienes tienen que relacionarse fluidamente para que se cumpla el cometido de la empresa.

Los procesos en una organización (que surge como resultado directo de los componentes de un sistema) se clasifican en primarios y secundarios:

- Proceso primarios: son los procesos que permiten la sostenibilidad y crecimiento del sistema organización. Todos los procesos dentro de la organización deben tener que ver con estos y deben soportarlos. De otra forma la organización incrementará su nivel de entropía interna.
- Procesos secundarios: son aquellos procesos que se realizan dentro de la organización para soportar a los procesos primarios [18]

El proceso de solicitud de compra de insumos de oficina, tomado como caso de estudio, pertenece al grupo de procesos secundarios, ya que es un proceso de apoyo, que dan soporte a los procesos operativos (ligados directamente con la realización del producto y/o a la prestación del servicio).

En el marco de una organización, los distintos departamentos de la misma pueden iniciar un proceso de solicitud de compra de insumos de oficina.

La organización modelo está compuesta por los siguientes departamentos:

- Dirección/Gerencia: decide los objetivos estratégicos a alcanzar por la empresa y los objetivos funcionales por cada departamento, supervisando y coordinando su cumplimiento, asignando recursos y presupuestos a cada uno.
- Control de Gestión: supervisa y vigila que todos los departamentos cumplan sus objetivos, reportando a la dirección general.
- Administración: contabiliza las facturas emitidas y recibidas, cobra a los clientes, paga a los proveedores y liquida impuestos en las fechas correspondientes.
- Comercial/ventas; consigue vender los objetivos de ventas planteados para que la empresa consiga una rentabilidad, atendiendo y fidelizando a los clientes.
- Marketing: colabora con el sector Comercial para conseguir más ventas y atender mejor a los clientes.
- Compras: adquiere materias primas e insumos para el buen funcionamiento de la organización.
- Recursos humanos: gestiona a los empleados de la empresa, decide fórmulas de contratación, remunera a los trabajadores y los mantiene motivados.
- Producción: fabrica el producto para luego ser comercializado por ventas.
- Finanzas: financiación de las necesidades de la empresa.

En la figura 28 se puede ver el organigrama de la organización.

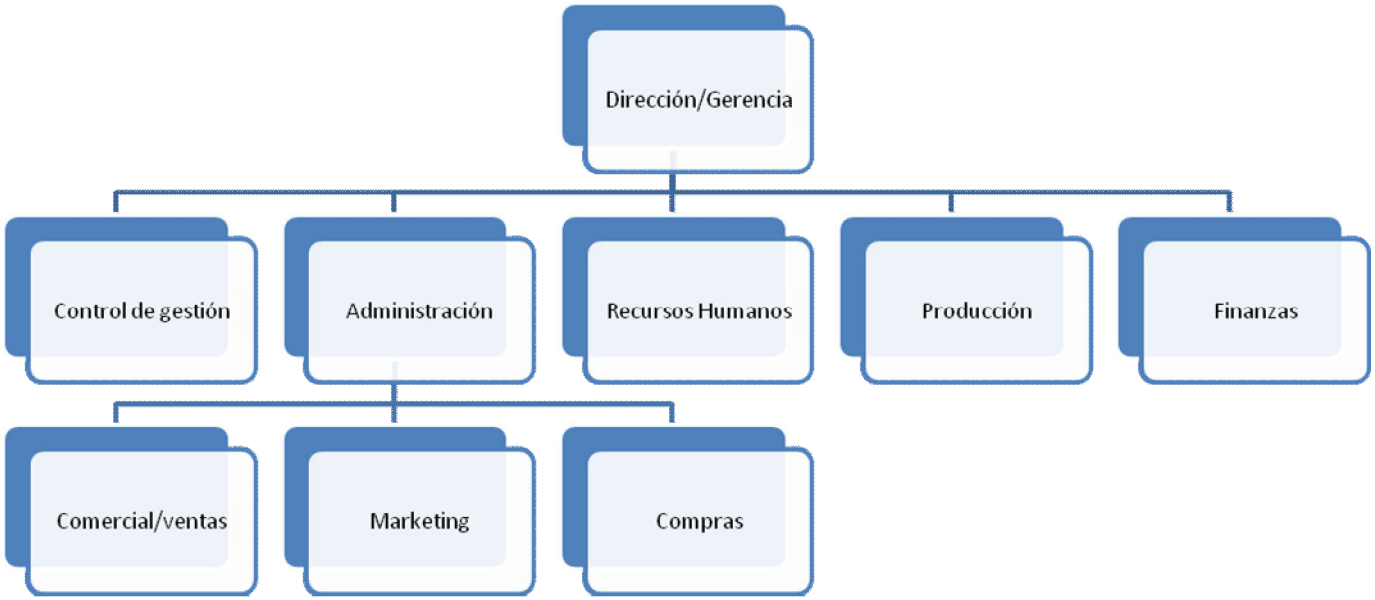


Figura 28 Organigrama de la organización modelo

El proceso de solicitud de compra se lleva a cabo en los departamentos de compras y administrativo. El proceso tiene tres tipos de roles cuyas funciones son las siguientes:

- Empleado Operario: encargado de cargar las solicitudes de compra de todos los departamentos
- Jefe: encargado de supervisar las solicitudes de compras realizadas por los empleados operarios.
- Empleado administrativo: encargado de gestionar el proceso de cotización de proveedores donde se piden las cotizaciones a los proveedores de los productos, y se selecciona a la cotización preferida. Luego genera la orden de compra y de ser necesario indica si la orden de compra requiere autorización por parte del gerente administrativo.
- Gerente administrativo: encargado de aprobar la orden de compra.

5.2 Diseño

En la figura 29 se puede visualizar el diagrama del proceso. Como se ve el proceso de solicitud de compra contiene dos subprocesos, el proceso de cotizaciones, y el proceso de orden de compra que son las dos últimas tareas que tienen el signo "+".

La primera tarea es llevada a cabo por un empleado operario, es una actividad manual llamada "Crear solicitud de compra". La siguiente tarea también es llevada a cabo por el empleado operario y se llama "Visualizar solicitud de compra", donde dicho empleado puede ver la solicitud y en caso de ser necesario corregirla. Luego la tarea "Autorizar solicitud" que es ejecutada por el jefe y decide si aprobar o no la solicitud. En caso de que la solicitud sea rechazada se le notifica al empleado operario en la tarea "Notificar rechazo", si la solicitud es aprobada se llama al subproceso "Solicitud cotizaciones" (Figura 30).

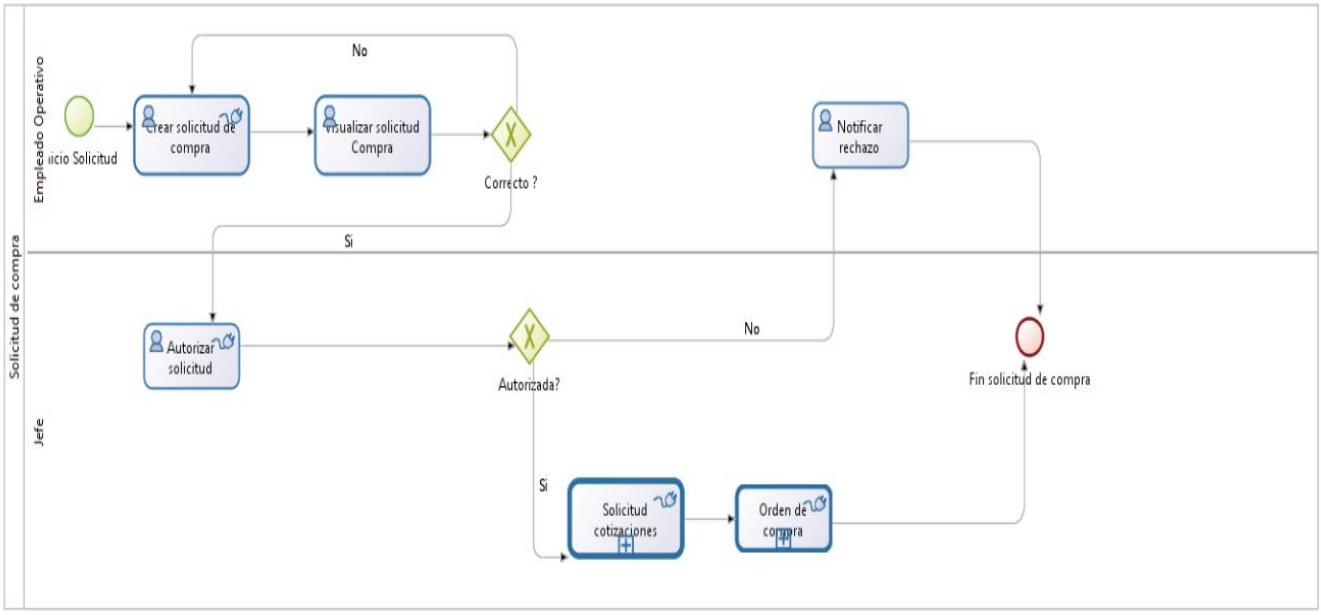


Figura 29 Proceso de solicitud de compra

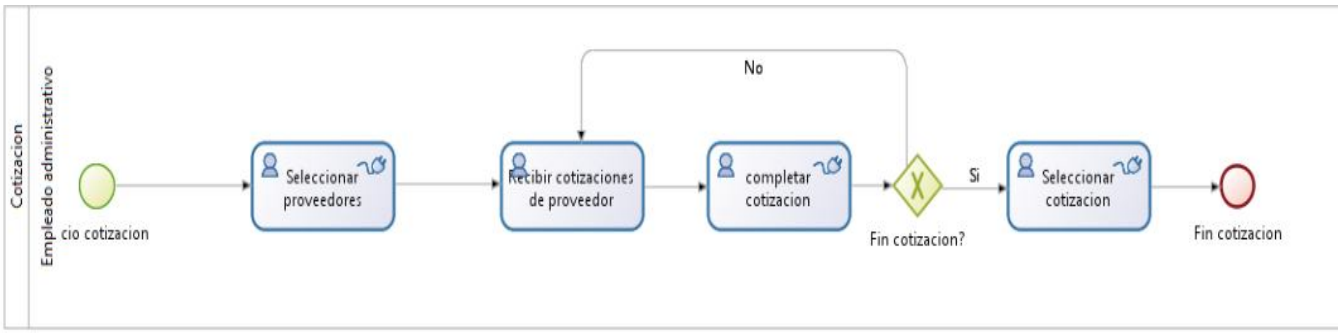


Figura 30 Subproceso solicitud cotizaciones

El subproceso de solicitud de cotizaciones está compuesto por cuatro tareas manuales llevadas a cabo por un empleado administrativo.

En la primera tarea, "Seleccionar proveedores", el empleado se encargará de seleccionar a los proveedores a los que desea pedir una cotización de los productos solicitados. Una vez seleccionados, el empleado debe notificar a los mismos por medio de un llamado telefónico. Cuando el proveedor se ponga en contacto para cotizar el producto, el empleado administrativo debe iniciar la tarea "Recibir cotizaciones de proveedor" donde selecciona el proveedor que va a cotizar, y en la siguiente tarea "Completar cotización", completa la cotización del mismo. Estas dos tareas se ejecutan para completar las cotizaciones de todos los proveedores. Una vez que se recibieron las mismas se selecciona una de ellas para efectuar la orden de compra.

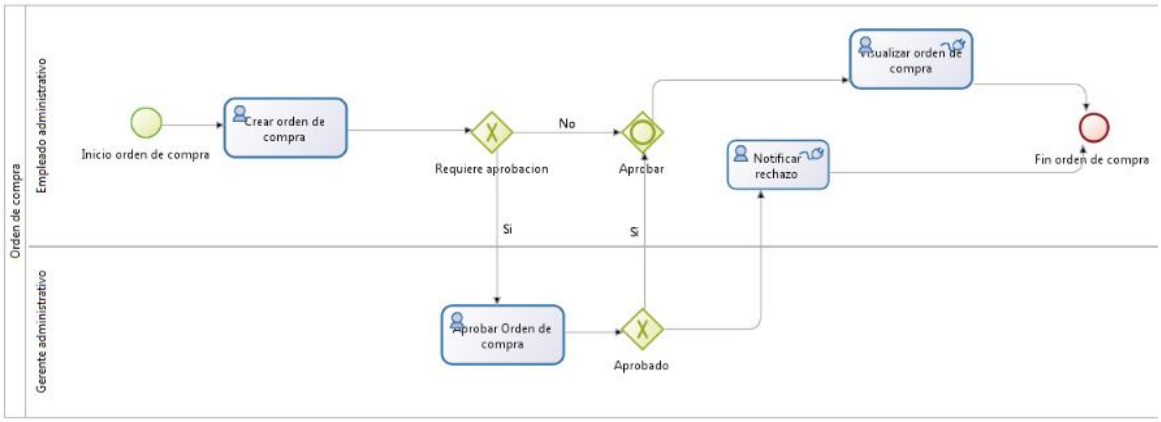


Figura 31 Subproceso orden de compra

En la figura 31 se ve el diseño del subproceso de Orden de compra. El mismo está compuesto por 4 actividades manuales. La primera actividad "Crear solicitud de compra" se lleva a cabo por el empelado administrativo y en el mismo se crea la solicitud de compra. Luego si requiere aprobación se ejecuta la actividad "Aprobar orden de compra" por el gerente administrativo, si no requiere aprobación o se aprobó la orden de compra se ejecuta la tarea "Visualizar orden de compra". Si el gerente administrativo rechaza la orden de compra se notifica el rechazo.

5.3 Implementación

El proceso de solicitud de compra se conecta constantemente con la base de datos del departamento de compras. El diagrama de la base de datos se ve en la figura 32, en dicha base se registran productos, proveedores, solicitudes de compra, cotizaciones y órdenes de compra

Para la implementación del proceso, se programó primero el proceso padre *solicitud de compra* y luego los subprocesos *solicitud de cotizaciones* y *orden de compra*.

Para cada uno de ellos se siguieron los siguientes pasos:

- Declaración de variables globales de proceso, que puede ser de los tipos prefinidos o puede ser un tipo Java complejo.
- Declaración de variables de tarea: que puede ser de los tipos prefinidos o puede ser un tipo Java complejo.
- Diseño de formularios o implementación de tarea automática: en este paso se diseña cada uno de los formularios de las tareas manuales o se codifican las tareas automáticas. En el diseño de formularios se debe conectar cada uno de los widgets de los mismos con las variables de tarea o proceso, y se permite agregar código java para procesar la visualización de cada uno de los widgets con código script Java.

En el anexo 4 se puede ver en detalle la implementación del proceso de solicitud de compra.

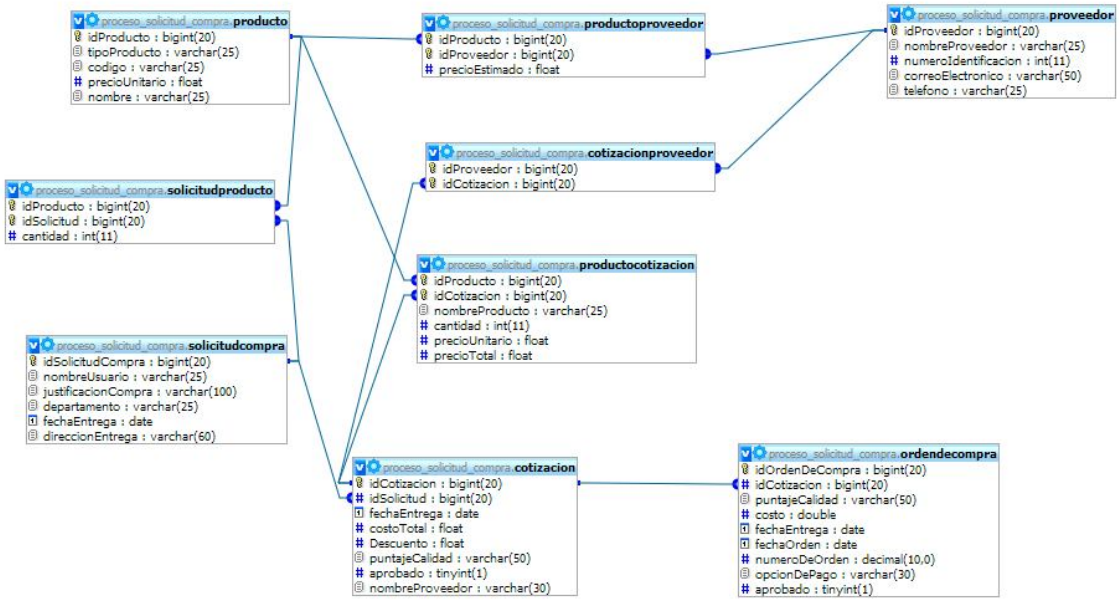


Figura 32 Diagrama de base de datos del proceso de solicitud de compra

5.4 Ejecución y monitoreo

En esta etapa se pone en producción el proceso implementado y se monitorea. Como Bonita Open Solution no cuenta con herramientas de análisis y monitoreo en su versión gratuita, utilizaremos el *proceso generador del log de eventos* implementado en el capítulo anterior, y luego el log de eventos generado será utilizado en la herramienta de

Process Mining ProM para aplicar las distintas técnicas de *Process Mining* que permiten el análisis y el monitoreo del proceso.

Para poder aplicar el proceso generador del log de eventos es necesario que se hayan ejecutado suficientes instancias de proceso de manera de que el mismo tenga un historial representativo, que permita realizar un buen análisis. En el caso de estudio el proceso estuvo en producción dos semanas, durante las cuales se ejecutaron un promedio de 3 instancias por día y luego se ejecutó el proceso generador del log de eventos. Se eligieron dos semanas y tres instancias por día por considerarse un flujo normal en la ejecución de un proceso de solicitud de compra en una organización mediana.

Los participantes que ejecutaron el proceso son:

- Empleados operarios: Martina Gerez con nombre de usuario mgerez y Analía Bora con nombre de usuario abora
- Jefe: Lorena Ramirez, con nombre de usuario lramirez
- Empleados administrativos: Andrea Lara con nombre de usuario alara y Fernando Bravo con nombre de usuario fbravo
- Gerente administrativo: Pablo Lansky con nombre de usuario plansky

A continuación se debe correr el proceso generador del log de eventos para el proceso de solicitud de compra, de cotizaciones y de orden de compra. Esto proceso generará tres logs de eventos. Llamados: logSolicitudDeCompra.xes, logCotizacion.xes y logOrdenDeCompra.xes.

En las siguientes secciones describiremos a grandes rasgos los beneficios en cuanto al análisis que nos brindan cada una de las técnicas seleccionadas: *View Inspector, Dotted chart, BPMN analysis*, sistema de transición *y Fuzzy miner*.

Luego en el anexo 5 se puede ver en detalle los resultados de la aplicación de las técnicas de *Process Mining* sobre el proceso de solicitud de compra y los subprocesos de solicitud de cotizaciones y orden de compra.

5.4.1 View Inspector

Al importar el log de eventos en la herramienta ProM, podemos visualizar el mismo con el *View Inspector* para obtener una visión general del mismo. En la figura 33 se pueden ver los diagramas de los eventos por caso, y debajo las clases de eventos por casos también. Como datos clave nos muestra la cantidad de procesos analizados, la cantidad de casos y la cantidad total de eventos que se observan, la cantidad de clases de eventos y la cantidad de tipos de eventos y finalmente la cantidad de originadores.

Nos permite también analizar en detalle cada una de las instancias de proceso, junto con las variables y los valores que tomaron cada una de ellas en cada evento.

Figura 33 *View Inspector* aplicado al proceso de solicitud de compra

En la pestaña *Inspector* del *View Inspector* se puede ver el detalle de todas las instancias de proceso y de cada uno de los eventos que la componen y los atributos tanto de las instancias de proceso como de los eventos. Figura 34.

Dentro del *Inspector,* en la pestaña explorer se pueden ver las instancias de proceso registradas en el log con gráfico de los eventos de cada una por color. Aquellos en verde son los eventos más frecuentes, y los colorados los menos frecuentes. Se puede deslizar el *mouse* por los eventos para ver más detalles de los mismos: fecha de ocurrencia del evento, participante que lo realizo y frecuencia de ocurrencia del mismo. En nuestro proceso de solicitud de compra, las tareas que se encuentran en naranja son las que pertenecen a los subprocesos y las de "Notificar rechazo", que por lo general son las tareas que más demoraron (Figura 34).

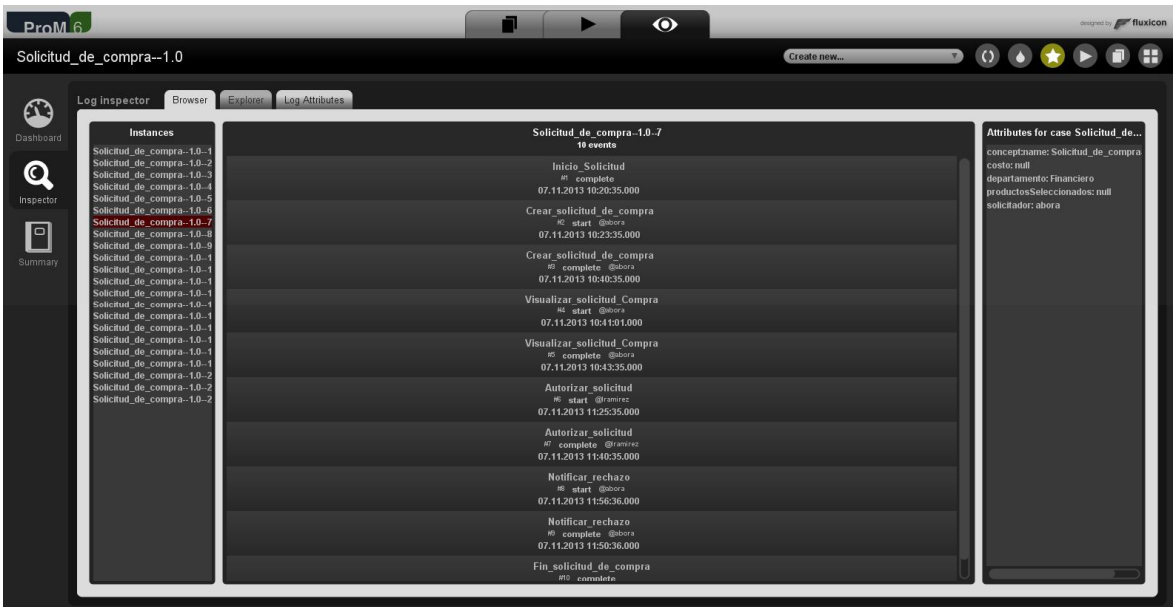


Figura 34 Browser aplicado al proceso de solicitud de compra

En la pestaña *log atributes* del *log inspector* se puede visualizar el detalle de los atributos de instancia y de evento como se ve en la figura 35.

Por ultimo en la pestaña *Summary* del *View Inspector* se poseen datos de análisis de proceso resumidos: eventos de proceso registrado cantidad de ocurrencias de cada uno y frecuencia de ocurrencia de los mismos. También los Eventos de inicio y de fin detectados, información de los participantes, frecuencia de participación y cantidad de veces que realizaron algún evento.

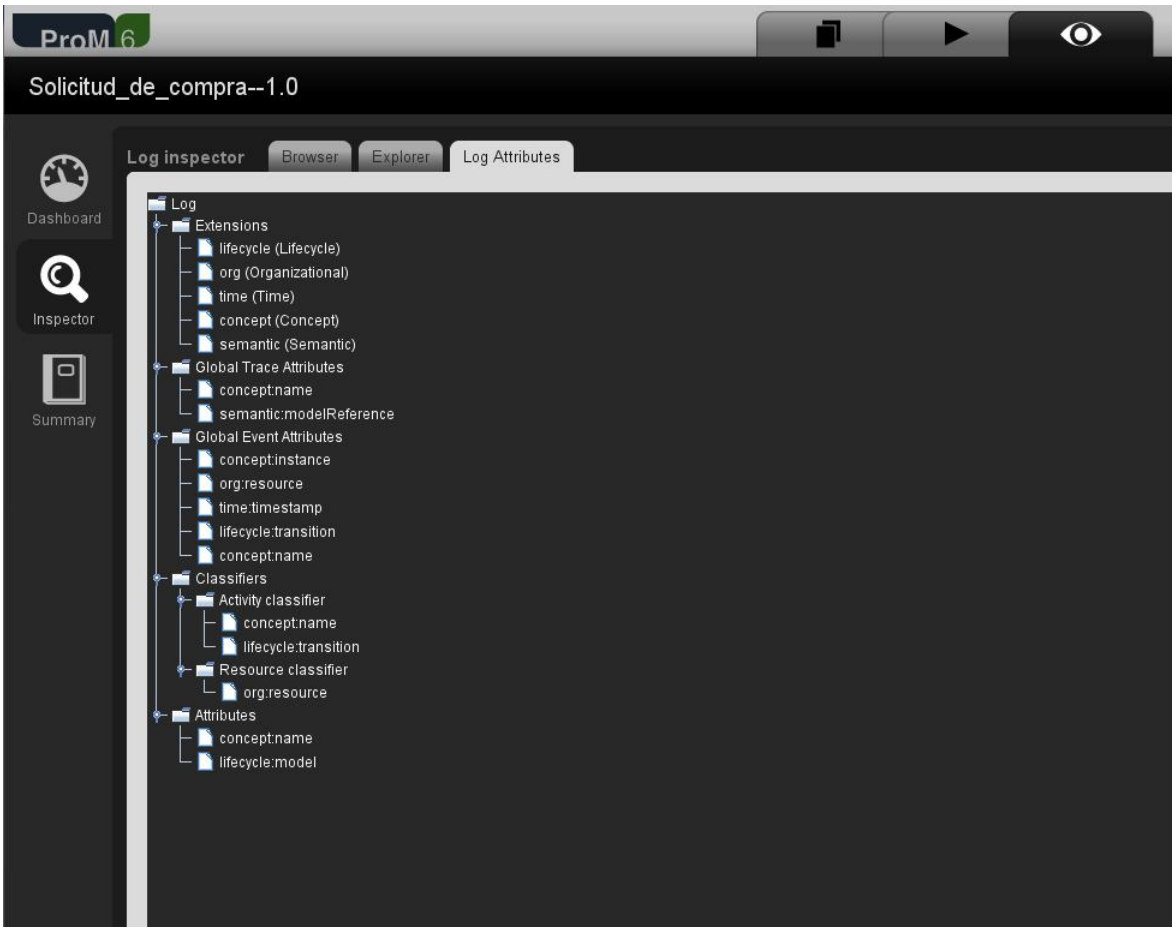


Figura 35 Log Attributes aplicado al proceso de solicitud de compra

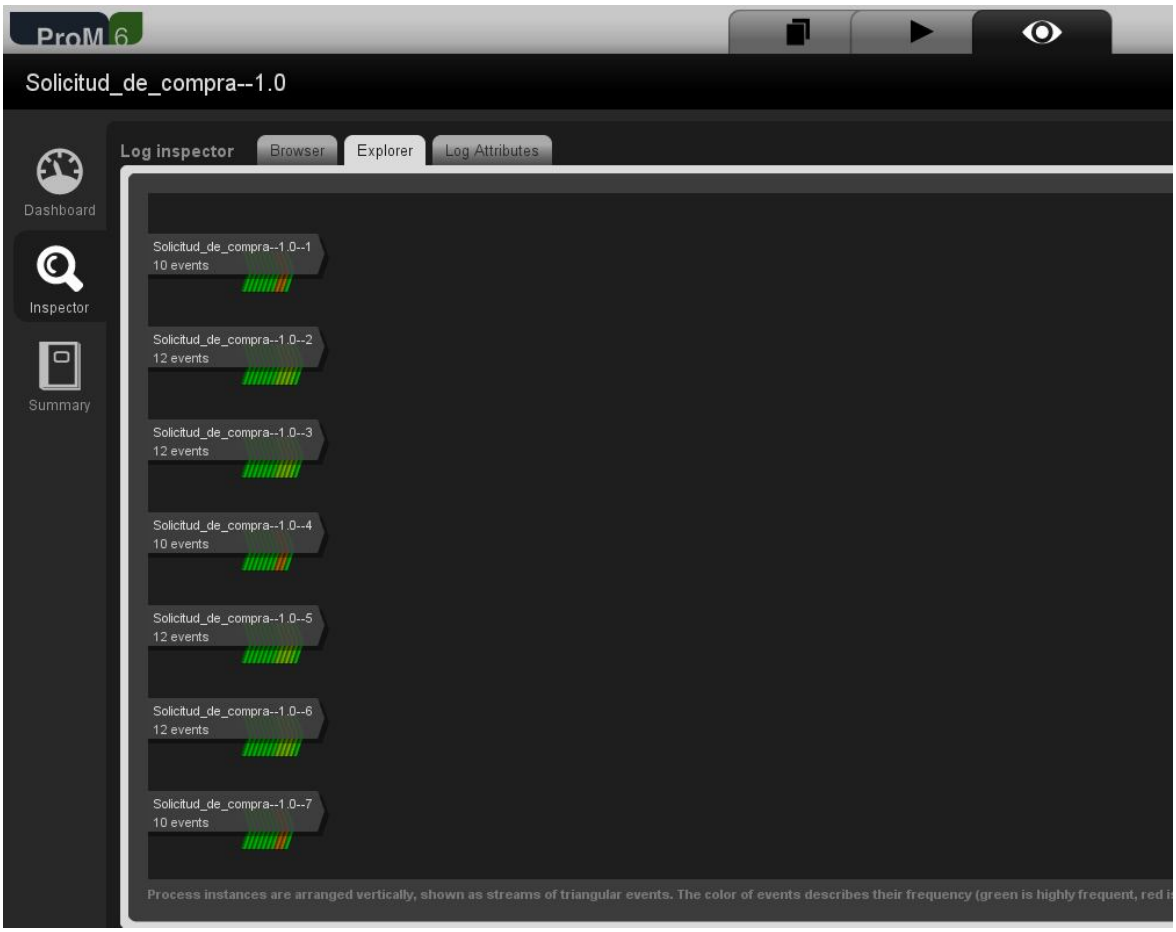


Figura 36 Explorer aplicado al proceso de solicitud de compra

Con el soporte de esta herramienta se puede importar el log de eventos y tener una visión general del proceso a analizar.

5.4.2 Dotted chart

Dotted chart es otra de las técnicas que permiten realizar un análisis sobre el procesos de negocio. Esta técnica permite visualizar información del proceso organizada sobre un gráfico de ejes X e Y, donde X representa el tiempo y el eje Y se puede configurar para mostrar instancias de proceso, instancias de actividad, participante, o evento.

Dotted chart permite evaluar el proceso en cuanto a las instancias de proceso, las tareas del proceso, los participantes y la performance en general.

Si aplicamos la técnica de *Dotted chart* sobre el proceso de solicitud de compra desplegando la instancia de proceso sobre el eje Y, se visualizará un gráfico como el de la figura 37. Los puntos representan la ocurrencia de las instancias de proceso en el tiempo. A la derecha de la pantalla se puede visualizar un detalle por instancia de proceso mostrando el tiempo de inicio y de fin de cada instancia.

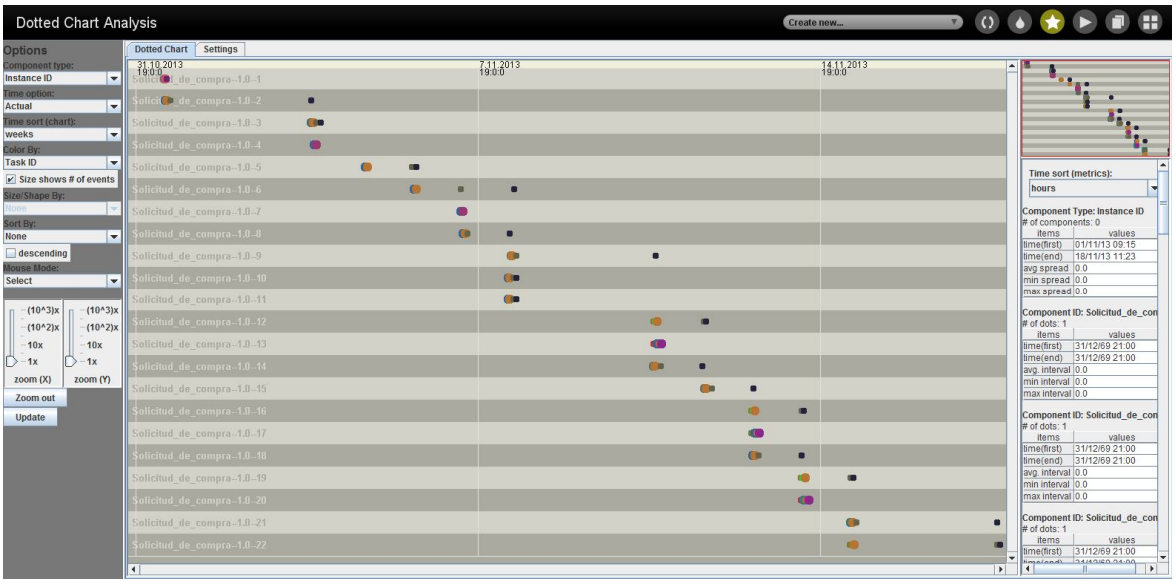


Figura 37 *Dotted chart* aplicado al proceso de solicitud de compra por instancia de proceso

5.4.3 BPMN analysis

La aplicación de la técnica de análisis BPMN permitirá visualizar el modelo en notación BPMN y nos permitirá hacer análisis de performance. También permite analizar las interacciones entre los participantes por medio de redes de trabajo social. Si se compara el proceso obtenido del análisis con el modelado en Bonita Open Solution, se puede ver que las actividades, flujos y compuertas se corresponden. Se puede observar también que los arcos varían su grosor, cuanto más fino sea el arco menos veces se toma ese camino mientras que cuanto más grueso sea, mayor frecuencia de ocurrencia tiene ese camino.

BPMN analysis permite identificar un número determinado de caminos más frecuentes. Por defecto identifica tres, y permite analizar y proyectar cada uno de estos por separado. A la derecha se tiene un apartado con la lista de caminos para proyección de información de performance donde puede verse los 3 caminos más frecuentes y con el botón "Project" proyectarlo.

En la figura 46 se puede ver la descripción de cada uno de los valores de las actividades desplegadas en el modelo. En una actividad se tiene información de:

- La probabilidad de la tarea en porcentaje
- La frecuencia de aparición de la actividad
- Tiempo total: si es verde es relativamente bajo, si es amarillo relativamente moderado y si es rojo relativamente alto.
- Tiempo de trabajo: si es verde es relativamente bajo, si es amarillo relativamente moderado y si es rojo relativamente alto.
- Tiempo de sincronización: si es verde es relativamente bajo, si es amarillo relativamente moderado y si es rojo relativamente alto.
- Tiempo de espera: si es verde es relativamente bajo, si es amarillo relativamente moderado y si es rojo relativamente alto.

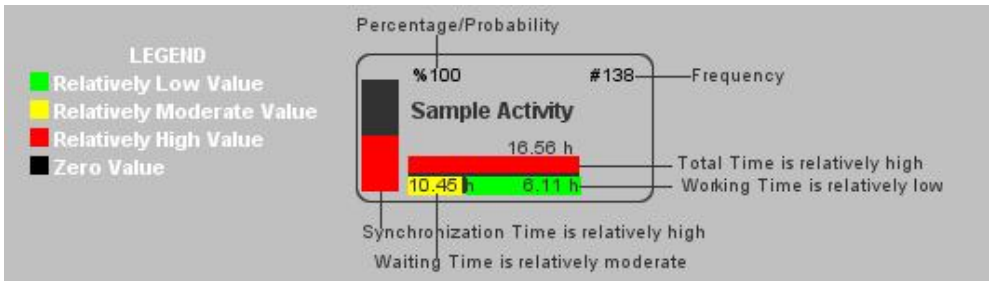


Figura 38 descripción de cada uno de los valores de las actividades *BPMN analysis*

Al costado del display se puede ver indicadores de performance y concordancia como se muestra en la figura 47.

Información de performance:

- Cantidad de casos
- Tasa de arribo por día
- Promedio de ejecución
- Tiempo mínimo de ejecución
- Tiempo máximo de ejecución

Información de concordancia

- Promedio de *fitness*
- *Fitness* menor
- *Fitness* Mayor

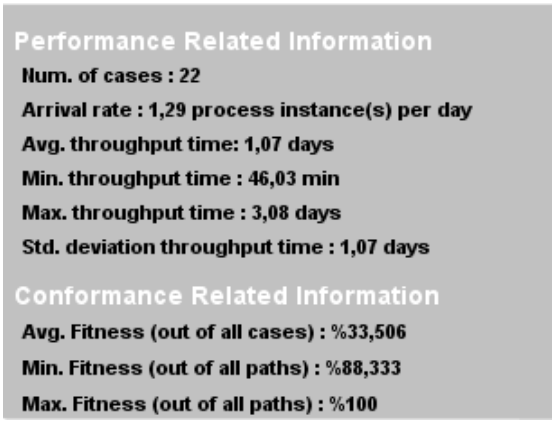


Figura 39 indicador de performance y de concordancia.

Si aplicamos la técnica de BPMN *analysis* sobre el log de Solicitud de compra, se visualizará el modelo en notación BPMN y nos permitirá hacer análisis de performance (figura 102). En la figura 48 también puede verse que los arcos varían su grosor. Por ejemplo el flujo de "Notificar rechazo" ocurre menos veces que el de "Solicitar cotizaciones" y "Orden de compra".

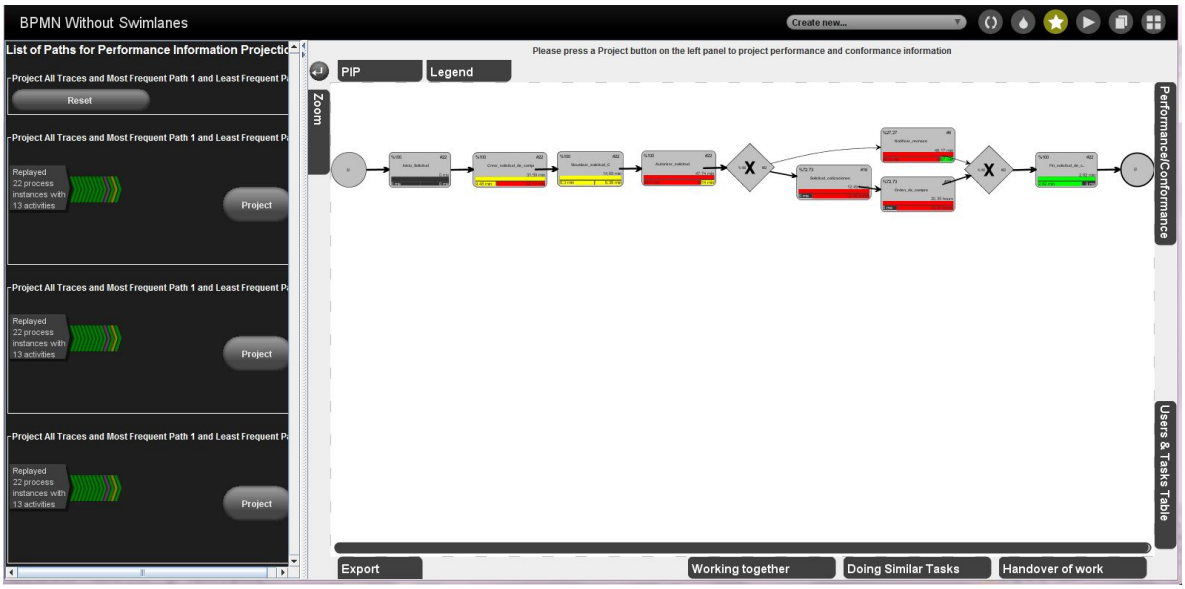


Figura 40 Aplicación de la técnica *BPMN analysis* sobre el proceso de solicitud de compra.

Como puede verse en el modelo, las tareas que más tiempo de trabajo tienen son la de solicitud de cotizaciones y orden de compra, siendo estos dos subprocesos. Estas dos actividades no tienen tiempo de espera porque son subprocesos que se inician automáticamente.

La tarea con menos tiempo de trabajo es la de "Notificar rechazo" pues en ella no debe hacerse más que leer la leyenda que informa el rechazo.

La tarea con mayor tiempo de espera es la de "Autorizar solicitud", y es aquí donde se puede producir un cuello de botella ya que se requiere de la interacción del participante Jefe.

Mediante las redes de trabajo se puede visualizar el pasaje de trabajo entre los participantes. En la figura 49 se puede ver perfectamente que el usuario mgerez solo toma contacto con lramiez, al igual que con abora, mientras que lramirez tiene contacto con los dos empleados administrativos.

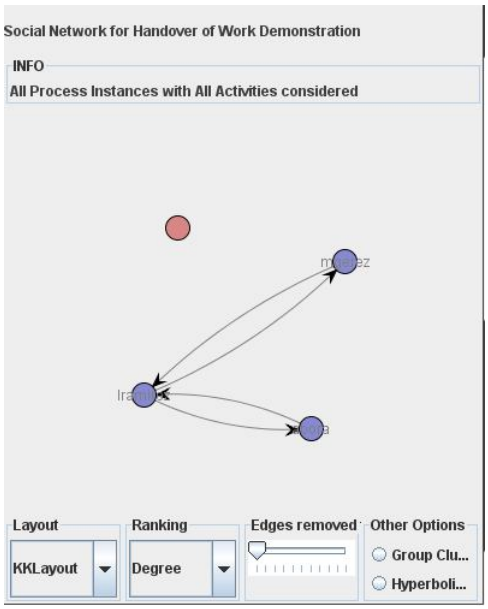


Figura 41 red de trabajo social de pasaje de trabajo entre los participantes

En la figura 50 se puede ver la red de trabajo social de los participantes que realizan tareas similares. Siendo los usuarios abora y mgerez los que realizan siempre las mismas tareas, pudiéndose así identificar dos roles entre los participantes, el rol de empleado operario y el de jefe.

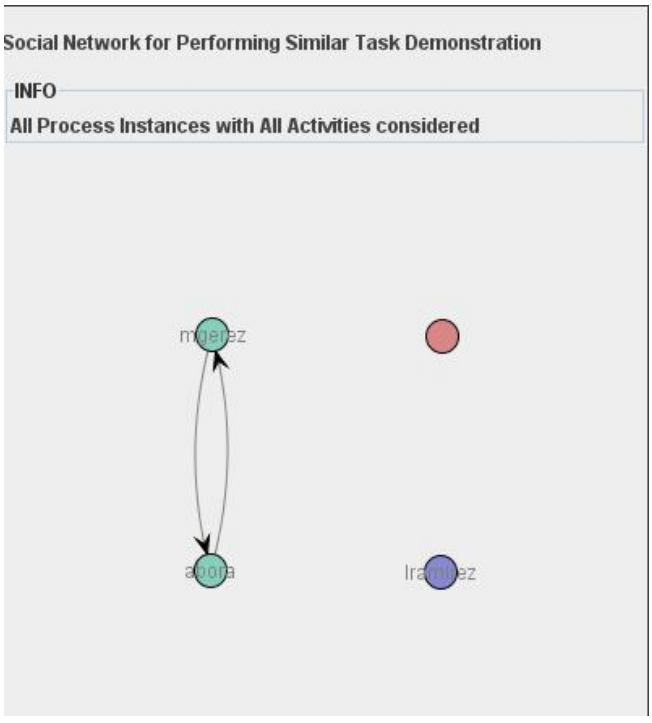


Figura 42 red de trabajo social de los participantes que realizan tareas similares.

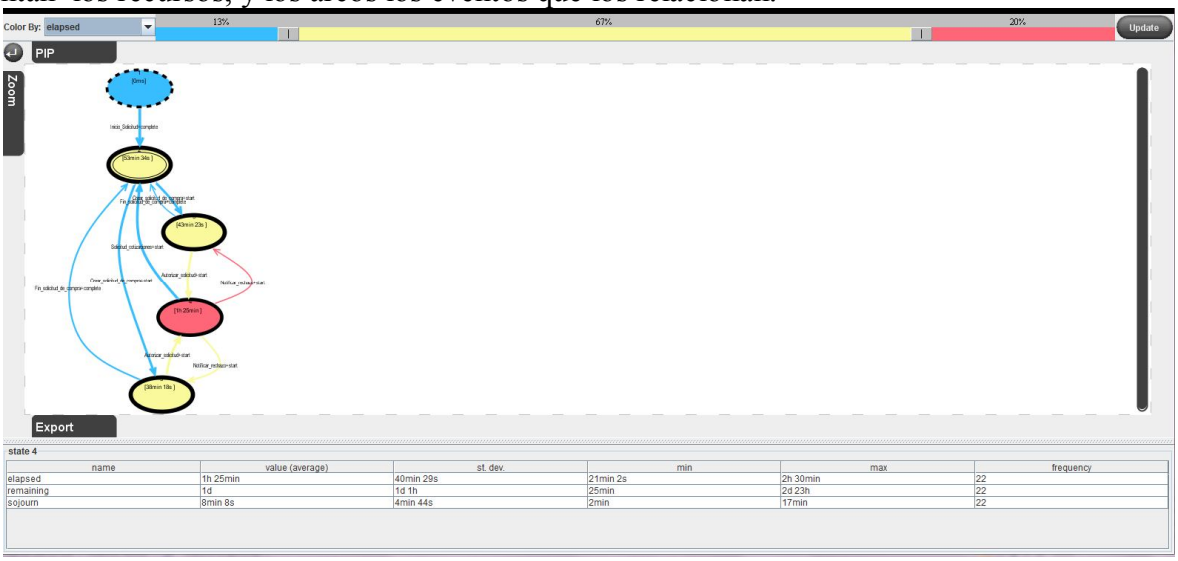
5.4.4 Sistema de transición

Mediante el sistema de transición se puede obtener diagramas de nodos y arcos donde los nodos pueden representar recursos, actividades, o una conjunción entre los mismos, mientras que los arcos representan los estados de las actividades.

Los colores de los arcos y de los nodos indican cuales fueron las tareas o recursos que más tiempo intervinieron en el proceso: azul menos tiempo, amarillo un tiempo mayor mientras que rojo es para las tareas críticas que más demoraron.

Por el grosor de los arcos se pueden determinar aquellas tareas o recursos más frecuentes.

En la figura 57 se puede apreciar un diagrama de transición donde los nodos representan los recursos, y los arcos los eventos que los relacionan.



name	value (average)	st. dev.	min	max	frequency
elapsed	1h 25min	40min 29s	21min 2s	2h 30min	22
remaining	1d	1d 1h	25min	2d 23h	22
sojourn	8min 8s	4min 44s	2min	17min	22

Figura 43 Diagrama de transición de estados del proceso de solicitud de compra.

Como puede verse el proceso de Solicitud de compra tiene cuatro recursos: los 3 participantes humanos, y se identifica un cuarto "participante" que es el automático (no tiene nombre). El color de los nodos indica cuál es el recurso que más tiempo tardó en completar la tarea. En este caso se puede ver que lramirez (el jefe) es el recurso que más tiempo tardó. El color y el grosor de los arcos indican también de los eventos cuales fueron los más frecuentes y cuál fue el evento que más tiempo demoró en completarse promedio. En nuestro ejemplo se ve que el inicio de la tarea "Notificar rechazo" fue el que más tiempo tardó en llevarse a cabo.

En la figura 58 se puede ver un diagrama de transición de estados donde los nodos representan a la dupla recurso-actividad mientras que los arcos representan a los eventos que los relacionan. Como puede verse hay un camino para cada recurso, al principio hay dos caminos (uno para abora y mgerez), que luego desembocan en lramirez. Luego se desprenden tres caminos, uno para el rechazo realizado por abora, otro para el rechazo de mgerez y finalmente el de aceptación donde se ejecutan los dos subprocesos de solicitud de cotizaciones y de orden de compra. Este camino no tiene recurso puesto ya que para nuestro proceso son tareas automáticas. Como puede verse mgerez tarda más que abora en completar la tarea de Notificar rechazo, mientras que la tarea que más tarda en completarse son la de orden de compra y solicitud de cotizaciones (los dos subprocesos). Esto puede verse por el color rojo en los arcos.

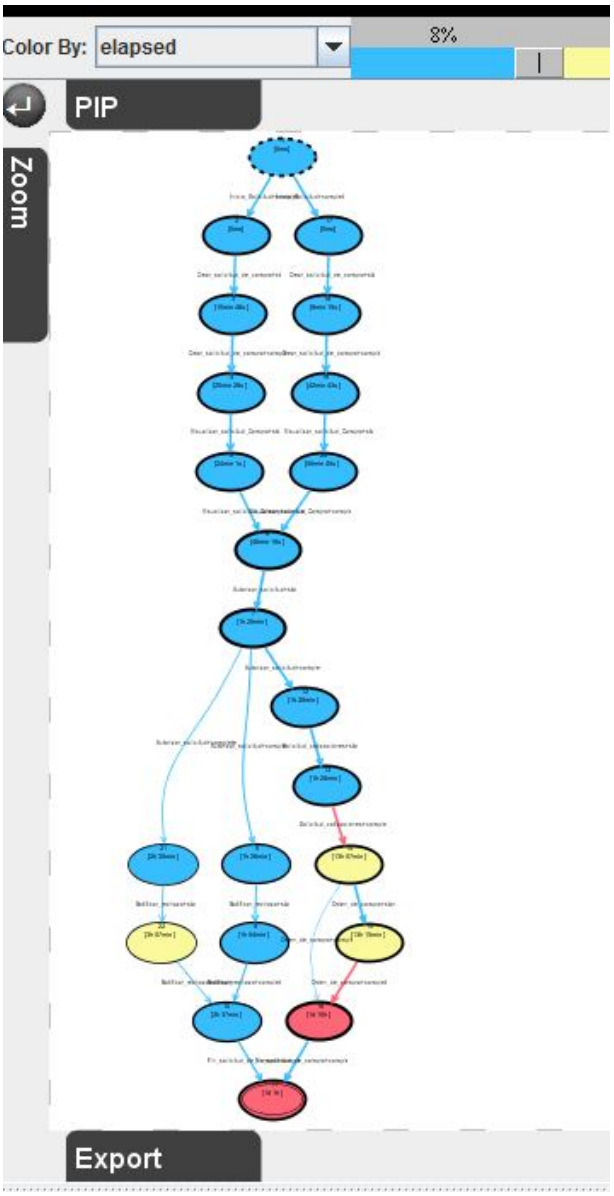


Figura 44 Diagrama de transición de estados del proceso de solicitud de compra

5.4.5 Fuzzy miner

Esta técnica permite producir un modelo fuzzy donde es posible visualizar las actividades más importantes de acuerdo a un grado de cauterización. También permite visualizar los flujos entre las actividades, a menor y mayor detalle. El grosor de los arcos nos indica la frecuencia de ocurrencia que tiene ese camino. La técnica de *Fuzzy miner* también permite aplicar una animación sobre una instancia de modelo *fuzzy* y la ejecución del log de eventos, permitiendo ver a simple vista la frecuencia de ejecución de las actividades y detectar posibles puntos de demora.

Si quisiéramos *clusterizar* las actividades según la frecuencia de utilización, obtendríamos un modelo como el de la figura 64, mientras que si quisiéramos ver con más detalle el flujo entre las actividades obtendríamos un modelo como el de la figura 65.

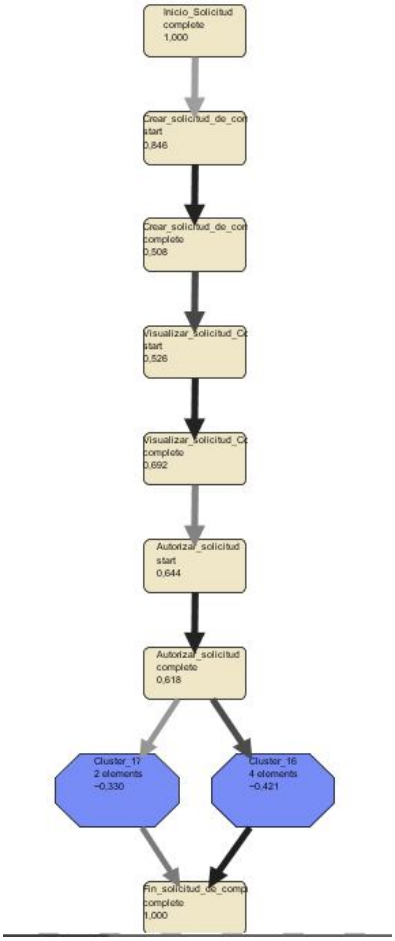


Figura 45 Modelo fuzzy aplicado al proceso de solicitud de compra clisterizando las actividades.

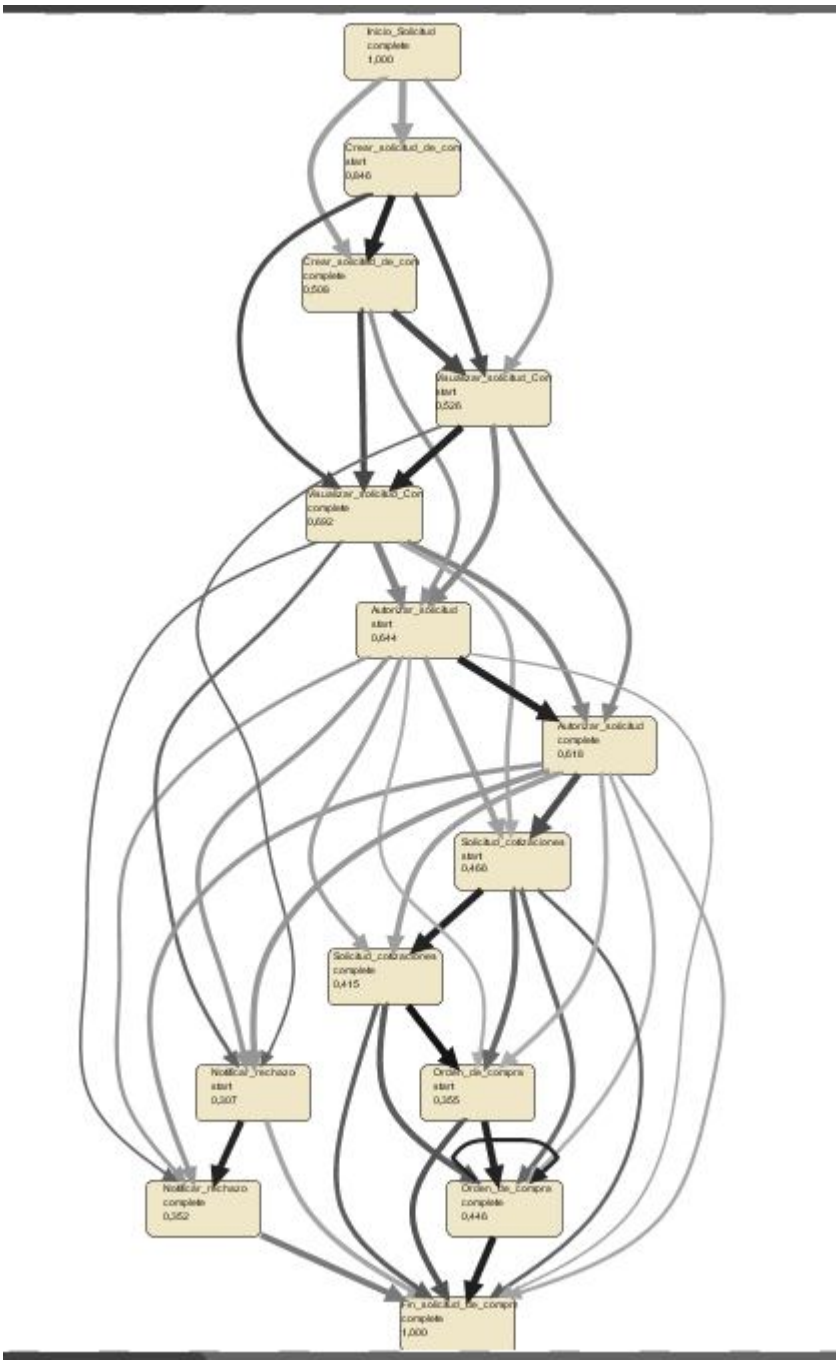


Figura 46 modelo *fuzzy* aplicado al proceso de solicitud de compra con más detalle de los caminos tomados.

Mediante la animación del modelo fuzzy se puede ver en nuestro proceso de solicitud de compra como el punto de demora está en la autorización de la solicitud y en los subprocesos de cotizaciones y orden de compra. En la figura 120 se muestra un ejemplo de la animación mientras corren los casos de la instancia de proceso.

Figura 47 Animación de un modelo fuzzy del proceso de solicitud de compra.

En resumen:

Estas son algunas de las técnicas de *Process Mining* que pueden asistirnos en el análisis y en la optimización de procesos de negocio.

Como puede verse las técnicas *View Inspector* y *Dotted Chart* nos ayudan a obtener una visión general del proceso, sus participantes, instancias, y grado de ocurrencia de cada uno de estos factores. Luego las técnicas de BPMN *analysis*, sistemas de transición, y *fuzzy minner* permiten analizar los flujos de ejecución del proceso, performance e interacción entre los participantes, ayudando a detectar cuellos de botella y deficiencias que pueden llevar a rediseñar el proceso, y comenzar con una nueva iteración en el ciclo de vida de BPM.

Conclusiones

En la actualidad las organizaciones usan tecnologías de información para soportar sus procesos de negocio siendo BPM (*Business Process Management* o Gestión de procesos de negocio) la tecnología pionera [1]. Para administrar los procesos de negocio las organizaciones utilizan BPMS donde se encuentran implementados los procesos de negocio.

Una de las etapas primordiales del ciclo de vida de BPM es la de análisis y optimización de procesos, en donde los procesos deben ser modificados/ rediseñados ya sea por la influencia de factores del entorno o una mala implementación del mismo. De acuerdo al modo de licencia del BPMS se puede contar con distintas herramientas de análisis y optimización, como BAM, BPI, etc., tecnologías bajo el paraguas de BI que suelen centrarse en los datos más que en el proceso.

Como vimos a lo largo del trabajo, *Process Mining* es una tecnología específica que se encuentra entre el proceso y los datos, capaz de asistir al ciclo de vida de BPM en todas sus etapas. Gracias a sus distintas técnicas y mediante el análisis de eventos complejos se puede alinear procesos con los requerimientos de cliente como cumplimiento, performance, calidad, eficiencia entre otros. [3] [4] [5] [6] [16].

Se puede utilizar *Process Mining* como metodología de análisis y optimización de procesos, siguiendo el enfoque tradicional para la aplicación de *Process Mining* (capitulo 3), pero este procedimiento puede requerir de un esfuerzo humano y de procesamiento costoso.

Sin embargo al encontrarse el proceso de negocio implantado en un BPMS nos encontramos ante un escenario en el cual:

- Ya se dispone de un modelo de proceso.
- Como el proceso ya se encuentra en producción, el mismo genera un historial en la base de datos del BPMS mismo, y este estaría compuesto por la instancia de proceso y las variables de la misma.
- La conexión de los eventos con las actividades del modelo se realiza innatamente por el BPMS, la base de datos del mismo almacena las instancias de procesos ejecutadas registrando para cada evento a que actividad del modelo corresponde.
- Un BPMS ya contempla las cuatro perspectivas: La perspectiva organizacional ya se encuentra relacionada al modelo porque cada actividad o *lane* debe tener un actor. En cuanto a la perspectiva de tiempo un BPMS siempre registra con *timestamps* la fecha de ocurrencia de todos los eventos. La perspectiva de caso está cubierta por las variables de proceso y de actividad que difieren de un caso a otro.

Considerando estas condiciones se definió un procedimiento para aplicar las técnicas de optimización de *Process Mining* a un proceso implantado en un BPMS:

- En una primera etapa se implementó el proceso en un BPMS con o sin asistencia de las técnicas de *Process Mining*, y ahora se quiere utilizar la misma para medir la performance, monitorear y optimizar el proceso.
- En la segunda etapa se debe tener un conocimiento de la ubicación de los datos en la base de datos del BPMS. Luego mediante una herramienta de generación de logs de eventos en formato XES extraen los datos y se genera el log.

- Una vez generado el log mediante una herramienta de *Process Mining* se puede importar el mismo y se le pueden aplicar las técnicas de optimización y monitoreo.

Por otro lado en la segunda etapa del enfoque propuesto las herramientas para la construcción del log de eventos son externas al BPMS y requieren de un conocimiento profundo de la ubicación de los datos en la base de datos del BPMS lo cual puede resultar complejo, ya que cada BPMS tiene su propia organización y administración de la base de datos [9] [11] [15], por este motivo surge interés en buscar una alternativa para la automatización de la extracción de logs de eventos desde dentro de un BPMS, agregándole funcionalidad al mismo y mejorando el nivel de integración del mismo.

Se seleccionó como herramienta de estudio Bonita Open Solution, por ser una herramienta de BPMS gratuita con una gran capacidad de extension gracias a su API implementada en Java y la capacidad de generar conectores con sistemas externos a medida.

Mediante la implementación de un proceso generador de log de eventos se automatiza la segunda etapa del enfoque propuesto para optimizar procesos implantados en un BPMS. Luego el log generado se utiliza como entrada en alguna herramienta de *Process Mining*, para este análisis se seleccionó Prom por su carácter de herramienta open source.

Finalmente mediante la aplicación de las distintas técnicas de *Process Mining*, se puede hacer un análisis y monitoreo exhaustivo del proceso, ayudando a evaluar los cambios a realizar o el rediseño del mismo, para así comenzar nuevamente con ciclo de vida de BPM.

Anexo 1 Data Mining base de las distintas técnicas de *Process Mining*

En este anexo se verán los orígenes de data mining sobre los que se basan las distintas técnicas de *Process Mining*.

Aprendizaje de árboles de decisión

Los árboles de decisión es una técnica de aprendizaje supervisado que tiene como objetivo la clasificación de instancias basada en variables predictivas. Hay una variable de respuesta categórica que etiqueta los datos y el resultado es organizado en la forma de un árbol. La Figura 48 muestra un ejemplo de la tabla descripta en la tabla 1.

Tabla 10 Conjunto de datos de 860 personas fallecidas recientemente, para estudiar los efectos del alcohol, el tabaco y el sobrepeso en la expectativa de vida [4]

Drinker	Smoker	Weight	Age
Yes	Yes	120	44
No	No	70	96
Yes	No	72	88
Yes	Yes	55	52
No	Yes	94	56
No	No	62	93
…	…	…	…

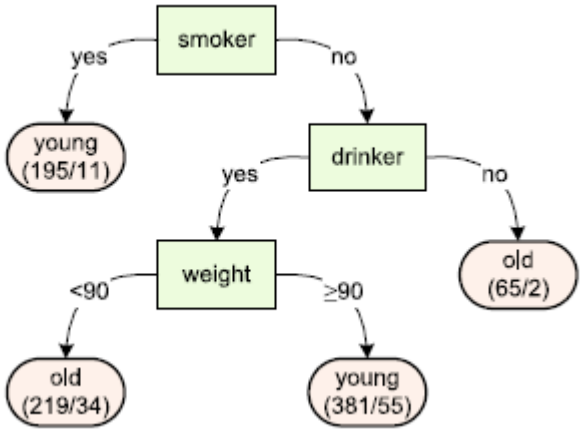


Figura 48 Un árbol de decisión derivado de la tabla 1. [4]

Las 860 personas se clasifican en jóvenes (si murieron antes de lo 70) en mayores (si murieron a los 70 o más tarde). Las personas que fuman por lo general mueren jóvenes (195 personas de las cuales 11 no clasifican). Las personas que no fuman y no beben tienen a vivir mucho más (65 personas de las cuales 2 no clasifican). Las personas que solo beben y además tienen sobrepeso (más de 90 kilos) también mueren jóvenes (381 personas de las cuales 55 no clasifican)

Las hojas corresponden a posibles valores de la variable de respuesta. Los nodos intermedios corresponden a variables predictivas. En el contexto de aprendizaje de árboles

de decisión, las variables predictivas se denominan atributos. Cada nodo atributo divide un conjunto de instancias en dos o más subconjuntos. El nodo raíz corresponde a todas las instancias.
Los arboles de decisión pueden ser obtenidos usando una variedad de técnicas. La mayoría de las técnicas utiliza un algoritmo top-down recursivo que funciona como sigue:

a. Crea un nodo raíz r y asocia todas las instancias al nodo raíz. $X := \{r\}$ es el conjunto de nodos a ser atravesado.

b. *Si* $X = \emptyset$, entonces retornar el árbol con la raíz r y terminar

c. Tomar todas las $x \in X$ y sacarlas de X, es decir, $X := X \setminus \{x\}$. Determina el puntaje *sold(x)* del nodo x antes de separar.

d. Determina si la separación es posible/necesaria. Si no, va al paso b. de otra forma se continua con el próximo paso.

e. Para todos los atributos posibles a ∈ A, evalúa los efectos de separar en el atributo. Toma a el atributo a que provea más mejoría, es decir maximice *snew a (x) − sold(x)*. El mismo atributo no debería aparecer múltiples veces en el mismo camino desde la raíz. Para atributos numéricos se tiene que determinar valores de corte.

f. Si la mejora es suficientemente sustancial, se crea un conjunto de nodos hijos Y, y se agrega Y a X y se conecta a x a todos los hijos de Y.

g. Asocia cada nodo en Y a su correspondiente conjunto de instancias *y* se va al paso *b*

Este es un esbozo del algoritmo genérico. Se deben realizar muchas decisiones de diseño para tener un aprendizaje de árboles de decisión concreto. Estas se pueden basar en la mejora de la función de puntaje o porque el árbol está restringido a una profundidad determinada. Hay muchas formas de seleccionar atributos. Estas pueden ser basadas en entropía, la diversidad del índice de Gini, etc. Cuando se selecciona un número de atributos para separar, se necesitan valores de corte que se determinen porque no es posible tener un nodo hijo para cada valor posible.
Un punto importante para observar es que por medio de la separación de un conjunto de instancias en subconjuntos es que la variación dentro de cada subconjunto se vuelve menor. Esto se puede ilustrar mejor utilizando la noción de entropía [4]

Entropía: codificación de incertidumbre

La entropía es una medida teórica de la información para la incertidumbre en un conjunto múltiple de elementos. Si el conjunto múltiple contiene muchos elementos diferentes y cada elemento es único, la variación es mínima y toma muchos bits codificar los elementos individuales. Por lo que la entropía es alta. Si todos los elementos en el conjunto múltiple de datos son iguales, entonces no se requieren bits para codificar los elementos individuales. En este caso la entropía es baja. Por ejemplo, la entropía de un conjunto múltiple *[A, B, C, D] rd* mucho más alta que la entropía de un conjunto múltiple *[A, A, A, A]* aunque cada conjunto tenga el mismo número de elementos.
Fórmula para calcular la entropía:

$$E = -\sum_{i=1}^{k} p_i \log_2 p_i$$

Aplicando el concepto de entropía a la figura del árbol de decisión anterior. Se etiqueta a las instancias en "old" o "young". Además por simplicidad nos abstraemos del atributo peso. En el paso inicial el árbol consiste únicamente de la raíz. Desde que la mayoría de las personas en nuestro conjunto de datos muere antes de los 70, etiquetamos este nodo como "young". Como de las 860 personas del conjunto de datos solo 546 mueren antes de los 70, las 314 personas restantes se encuentran descalificadas. Calculemos la entropía de ese nodo:

E = −(((546/860) log2(546/860)) + ((314/860) log2(314/860))) = 0.946848.

Este valor está muy cercano al valor máximo de uno.

En la Figura 49 de abajo se muestra el cálculo de la entropía en cada uno de los nodos y se observa como la misma va disminuyendo a medida que los nodos se siguen sub

clasificando

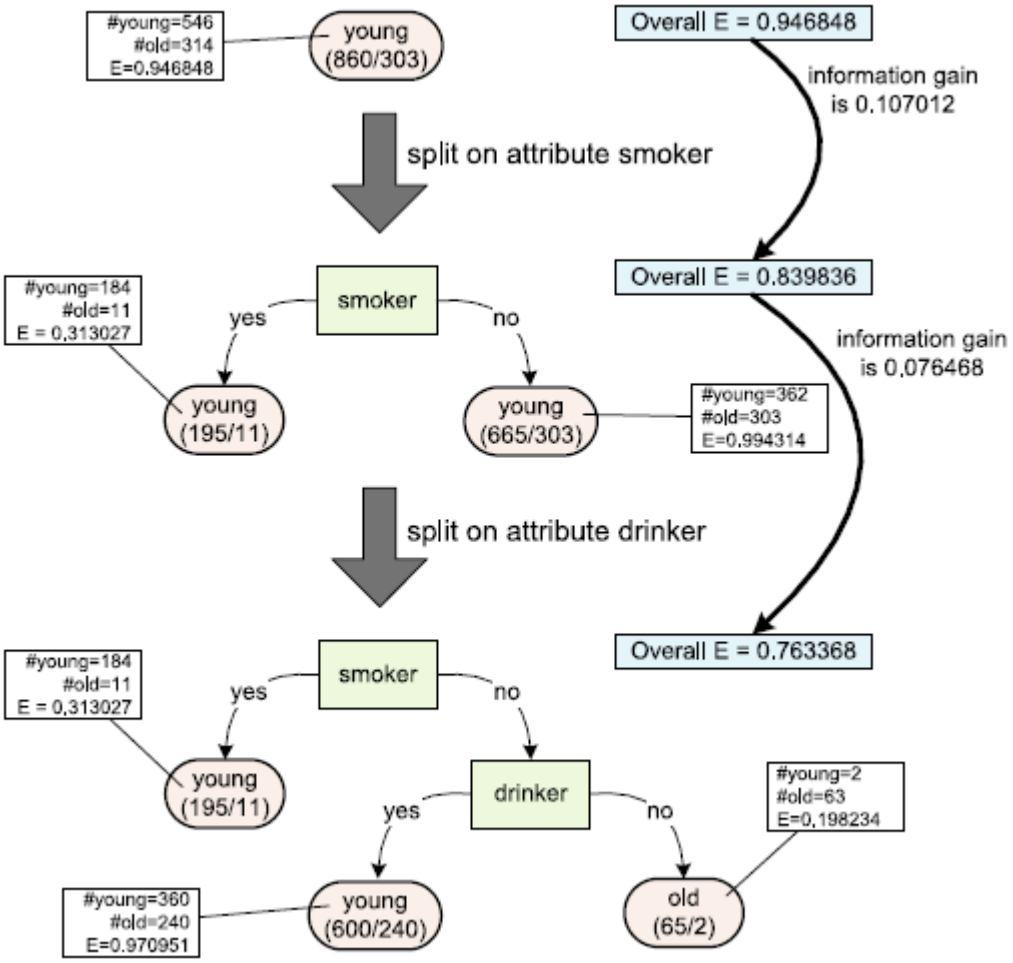


Figura 49 Construcción paso a paso del árbol de decisión obtenido por la información ganada basándose en entropía. [4]

La separación de los nodos siempre reducirá la entropía global. En el caso extremo de que todos los nodos hojas correspondan a un único individuo (o individuos que tengan exactamente los mismos valores de atributos). La entropía general es 0. Sin embargo, el árbol resultante no es muy útil y probablemente tenga pocos valores predictivos. Es vital darse cuenta que los arboles de decisión se basan en ejemplos. Un árbol de decisión que este sobre ajustado es complejo y no realiza buena utilización de las instancias no utilizadas. Por esto, es importante seleccionar los atributos correctos y dejar de separar cuando el menor sea ganado.
Gini index of diversity es otra medida para la diversidad de los nodos hojas. *Gini index of diversity* mide la "impureza" del conjunto de datos*.: G= 2- sumatoria i=1 a k de (pi*). Si todas las clasificaciones son iguales, entonces *G=0.* Si G se aproxima a 1 si hay más diversidad. Por esto, un enfoque es tomar un atributo que maximice la reducción del valor de G (en vez del de entropía).
El aprendizaje de árboles de decisión no está relacionado con el descubrimiento de procesos, sin embargo se puede usar en combinación con técnicas de *Process Mining*. Por

ejemplo, las técnicas de descubrimiento de procesos, como el algoritmo α ayudan a localizar todos los puntos de decisión en el proceso (XOR/OR). Subsecuentemente podemos analizar cada punto de decisión utilizando aprendizaje de decisiones. La variable respuesta es el camino tomado y los atributos y los elementos de datos se conocen de antes o en el momento del punto de decisión. [4]

K-means clustering

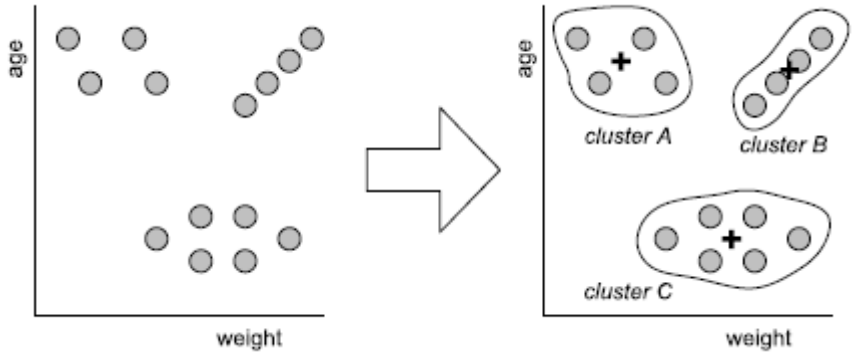
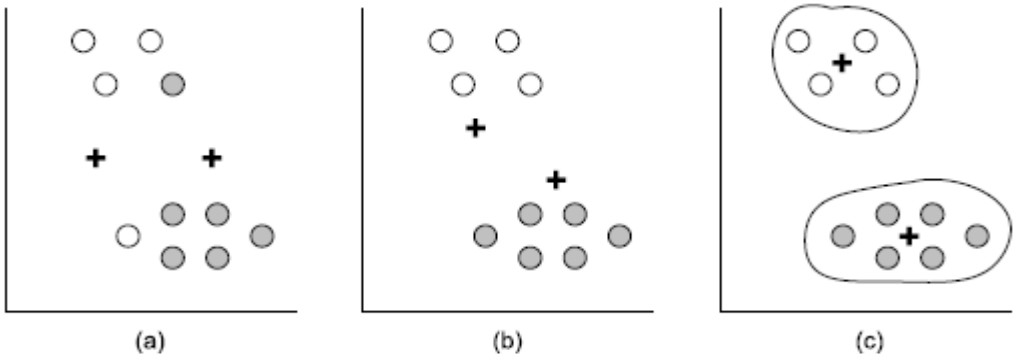


Figura 50 Instancias de Clustering en tres clústeres usando la técnica K-mean [4]

Las técnicas de clustering se ocupan de la agrupación de instancias en casos. Las instancias en un clúster deben ser similares unas con otras y no similares a instancias en otros clústeres. Clustering utiliza datos no etiquetados, por lo que, requiere una técnica de aprendizaje no supervisado.
Existen muchos algoritmos de clustering. Aquí nos focalizaremos en k-means clustering. Se puede observar en la Figura 50 una idea básica de clustering. Asumimos que tenemos un conjunto de datos con dos variables peso y edad. Los puntos corresponden a personas que tienen una edad y un peso particular. A través de una técnica de clustering k-means se puede descubrir los tres clústeres mostrados en la derecha. Idealmente, las instancias en un clúster se encuentran cerca unas de las otras mientras que estas se encuentran lejos de las que están en otros clústeres. Cada clúster tiene un centroide denotado con +. El centroide denota el centro del clúster y puede ser computado sacando el promedio de las coordenadas de las instancias en el clúster. Puede haber más de dos dimensiones en el clúster, en la figura anterior tenemos solo dos dimensiones el peso y la edad.
Los algoritmos de clustering basados en distancias como k-means y clustering jerárquico aglomerativo asumen una noción de distancia. El enfoque más común es considerar a cada instancia como un vector con n-dimensiones donde n es el número de variables y luego se calcula la distancia *Euclidian.* Para este propósito los valores ordinales y también los binarios se tienen que convertir en numéricos. La escala es importante cuando se define una métrica de distancia.
En la siguiente figura se puede ilustrar la idea básica de clustering k-means. El enfoque comienza con una inicialización random de dos centroides que se denotan con el símbolo +. En la Figura *a* los centroides se colocan aleatoriamente en el espacio de las dos dimensiones. Utilizando la métrica de distancia seleccionada, todas las instancias son asignadas al centroide más próximo. Aquí utilizamos la distancia estándar de *Euclidian*. Todas las instancias con un punto abierto son asignadas al centroide de la izquierda

mientras que todas las instancias con un punto cerrado son asignadas al centroide de la derecha. Basándonos en esta asignación tenemos dos clusters iniciales. Luego se computa el centro real de cada clúster. Finalmente volvemos a asignar las instancias al centroide que se encuentre más próximo. Luego de converger la salida del algoritmo k-means tenemos dos clusters y estadísticas relacionadas.



La calidad de un clúster particular se puede definir como la distancia promedio desde una instancia y su centroide correspondiente. K-means clustering es solo heurístico y no garantiza que se encontraran los k clusters que minimicen la distancia promedio desde una instancia a su centroide correspondiente. En realidad, el resultado depende de la inicialización. Por lo tanto, es bueno ejecutar repetidas veces el algoritmo con diferentes inicializaciones y seleccionar la mejor.
Uno de los problemas de utilizar k-means es determinar el número de clusters *k*. Para k-means esto se arregla desde el comienzo. Notar que la distancia promedio de una instancia a su centroide correspondiente decrece a medida que *k* crece. En el caso extremo cada instancia tiene su propio clúster y la distancia promedio desde una instancia a su centroide correspondiente es cero. Esto no es muy útil. Por esto, un enfoque frecuente es comenzar con un número pequeño de clusters y luego gradualmente incrementar k mientras haya mejoras significantes.
Otra técnica popular de clustering es el clustering jerárquico aglomerativo (AHC). Con este algoritmo un número variable de clusters se generan. El enfoque funciona de la siguiente manera:
Asigna cada instancia a un clúster singleton dedicado. Luego busca aquellos clusters que se encuentran más ceca uno del otro. Mezcla estos dos clusters en un nuevo clúster. Luego busca de nuevo dos clusters que se encuentren uno cerca del otro y los mezcla. Esto se puede repetir hasta que todas las instancias están en el mismo clúster. Se puede visualizar los clúster en un dendograma como muestra la figura b.

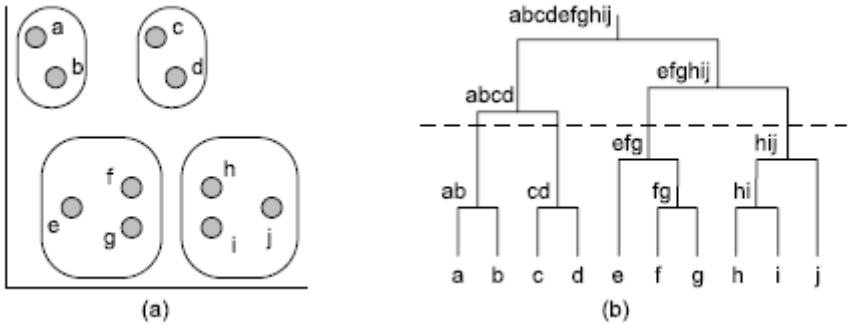


Figura 51 Cualquier línea horizontal en el dendograma corresponde a un clúster concreto en un nivel particular de abstracción [4]

Cada línea horizontal del dendrograma corresponde a un clúster concreto. Moviendo la

línea hacia arriba se llega a un solo clúster que contiene todas las instancias. Moviendo la línea horizontal el usuario puede variar el nivel de abstracción.
Las técnicas de clustering se encuentran indirectamente relacionadas con el descubrimiento de proceso. Sin embargo clustering se puede usar como un paso de pre procesamiento para *Process Mining*. Por medio de la agrupación de casos similares se pueden construir modelos de procesos parciales que son más fáciles de entender, por lo que puede ser útil primero identificar clusters y luego descubrir modelos más simples por clúster. [4]

Aprendizaje de reglas de asociación

Los árboles de decisión se pueden usar para predecir los valores de algunas variables respuesta que se han identificado como importantes. Conducidos por las variables respuesta, reglas como "las personas que toman y fuman mueren antes de los 70" se pueden encontrar. El objetivo del aprendizaje de reglas de asociación es encontrar reglas similares pero ahora sin focalizarse en una variable de respuesta particular. El objetivo es encontrar reglas como *If X THEN Y* donde *X* es llamado antecedente *e Y* consecuente. Esas reglas también se denotan *como* $X \Rightarrow Y$. *X* e *Y* pueden ser cualquier conjunción de "variable= valor". La única condición es que *X* e *Y* no sean vacías y que cualquier variable aparezca como mucho una vez en *X* e *Y*.
Ejemplo IF smoker = no AND age ≥ 70 THEN drinker = yes. Por lo general se consideran sólo variables categóricas. Sin embargo, hay varias técnicas para transformar variables numéricas en variables categóricas.
Cuando se aprenden reglas de asociación de la expresión $X \Rightarrow Y$, se usan tres métricas: soporte, confidencia y elevación. Sea *Nx* el número de instancias para los que *X* se sostiene. *Ny* es el número de instancias para los que *Y* se sostiene. $NX \wedge Y$ es el número de instancias para los que ambas *X* e *Y* se sostiene. N es el número total de instancias. El soporte de una regla $X \Rightarrow Y$ se define como:
soporte $(X \Rightarrow Y) = NX \wedge Y / N$
El soporte indica la aplicabilidad del enfoque, es decir, la fracción de instancias que sostienen el antecedente y el consecuente. Por lo general una regla de alto soporte es más útil que una regla con bajo soporte.
La confidencia de una regla $X \Rightarrow Y$ se define como:
confidencia $(X \Rightarrow Y) = NX \wedge Y / NX$
Una regla con alta confidencia es decir un valor cercano a 1, indica que la regla es muy confiable, es decir que si *X* sostiene, entonces *Y* también sostendrá. Una regla con alta confidencia es definitivamente más útil que una regla con baja confidencie.
La elevación (lift) de una regla $X \Rightarrow Y$ se definen como:

$$lift(X \Rightarrow Y) = \frac{N_{X \wedge Y}/N}{(N_X/N)\,(N_Y/N)} = \frac{N_{X \wedge Y}\,N}{N_X\,N_Y}$$

Si *X* e *Y* son independientes, entonces la elevación estará cerca de 1. Si el *lift* $(X \Rightarrow Y) > 1$, entonces *X* e *Y* correlacionan positivamente. Por ejemplo si $lift(X \Rightarrow Y) = 5$ quiere decir que *X* e *Y* ocurren más veces juntas que si fueran independientes. Si *ift* $(X \Rightarrow Y) < 1$, entonces *X* e *Y* correlacionan negativamente (es decir, la ocurrencia de *X* hace que no ocurra tantas veces *Y* y viceversa). Las reglas con un alto valor de elevación son por lo general más interesantes. Sin embargo, los valores de elevación son sólo considerados si se cumplen ciertos umbrales con respecto al soporte y a la confidencia.

Una de las técnicas que explica el aprendizaje de reglas de asociación es *market basket analysis*. En esta técnica solo consideramos variables binarias que pueden ser interpretadas como presente o no. Como se ve en la Tabla 2.

Tabla 11 Conjunto de datos de 240 órdenes de clientes en una cafetería registrado por el registrador de cambio [4]

Cappuccino	Latte	Espresso	Americano	Ristretto	Tea	Muffin	Bagel
1	0	0	0	0	0	1	0
0	2	0	0	0	0	1	1
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	1	2	0
0	0	0	1	1	0	0	0
...	...	...	...	...	...	...	...

Si ignoramos el número de ítems ordenados, se puede sobrescribir el conjunto de datos en términos de conjunto de ítems [{cappuccino, muffin}, {latte, muffin, bagel}, {espresso}, {cappuccino}, {tea, muffin}, {americano, ristretto},. . .]. El "*market basket" analysis* analiza esa entrada sistemáticamente. Basándose en el conjunto de ítems, el objetivo es generar reglas con la forma $X \Rightarrow Y$ donde X e Y se refieren a conjuntos no vacíos disjuntos de ítems.
En la tabla hay $N=240$ clientes. $Ntea= 50$, $Nlatte=40$, $Nmuffin=40$, $Ntea^late=20$, y $Ntea \wedge latte \wedge muffin = 15$ (ordenes que incluyen al menos una taza de té, de latte y un muffin). Dados los números computados fácilmente se puede obtener las tres métricas definidas previamente.

$$support(X \Rightarrow Y) = N_{X \wedge Y}/N = N_{tea \wedge latte \wedge muffin}/N = 15/240 = 0.0625$$

$$confidence(X \Rightarrow Y) = N_{X \wedge Y}/N_X = N_{tea \wedge latte \wedge muffin}/N_{tea \wedge latte} = 15/20 = 0.75$$

$$lift(X \Rightarrow Y) = \frac{N_{X \wedge Y}\ N}{N_X\ N_Y} = \frac{N_{tea \wedge latte \wedge muffin}\ N}{N_{tea \wedge latte}\ N_{muffin}} = \frac{15 \times 240}{20 \times 40} = 4.5$$

Para generar reglas de asociación sistemáticamente, se definen por lo general dos parámetros*: minsup* y *minconf*. El soporte de cualquier regla $X \Rightarrow Y$ debe estar debajo del umbral *minsup* es decir, $support(X \Rightarrow Y) \geq minsup$. De igual forma la confidencia de cualquier regla tiene que estar debajo del umbral *minconf*, es decir, $confidence(X \Rightarrow Y) \geq minconf$.
Las reglas de asociación se generan de la siguiente forma:

1. Se generan todos los conjuntos de ítems frecuentes, todos los conjuntos Z tal que $NZ/N \geq minsup$ y $|Z| \geq 2$.
2. Para cada conjunto de ítems frecuentes en Z, considerar todas las particiones de Z en dos subconjuntos no vacíos X e Y. Si $confidence\ (X \Rightarrow Y) \geq minconf$, entonces se toma la regla $X \Rightarrow Y$. Si $confidence(X \Rightarrow Y) < minconf$ entonces descartar la regla
3. La salida son las reglas encontradas.

Este algoritmo simple tiene dos problemas. Primero, hay un problema computacional relacionado con el primer paso. Si hay *m* variables entonces hay *2 a la m - m-1* conjunto de ítems posibles. Por lo tanto para 100 productor *(m=100)* hay *1267650600228229401496703205275* conjuntos de ítems frecuentes candidatos. El segundo problema es se generan muchas reglas que no interesan. Por ejemplo luego de presentar la regla *tea ∧ latte⇒muffin*, no tiene sentido mostrar la regla *tea⇒latte ∧ muffin* aunque el *minsup* y el *minconf* estén en el umbral. Muchas técnicas se han desarrollado para mejorar el seed-up en la generación de reglas de asociación y seleccionar las más interesantes. Una de estas el algoritmo Apriori.
Las reglas de asociación están relacionadas con el descubrimiento de procesos. Además el *α-algorithm* atraviesa el log de eventos buscando patrones. Sin embargo, las reglas asociativas no consideran el orden de las actividades, y no tienen como objetivo construir un modelo de proceso total. [4]

A.1.1 Minería de episodio y de secuencia

El algoritmo Apriori usa la propiedad de la monotonicidad en donde todos los subconjuntos de un conjunto de ítems frecuentes también son frecuentes. Muchos problemas de descubrimiento de patrones o reglas tienen las mismas propiedades de monotonicidad, así se habilitan implementaciones eficientes. Un ejemplo conocido es la minería de patrones secuenciales.

Minería de secuencia

El algoritmo de Apriori no considera el orden de los eventos. La minería de secuencia soluciona este problema analizando secuencias del conjunto de ítems. Uno de los primeros enfoques fue realizado por Srikant y Agrawal. Para ilustrar la minería de secuencia utilizaremos como ejemplo la Tabla 3.

Tabla 12 Un fragmento de un conjunto de datos para minería de secuencia: cada línea corresponde a una orden [4]

Customer	Seq. number	Timestamp	Items
Wil	1	02-01-2011:09.02	{*cappuccino*}
	2	03-01-2011:10.06	{*espresso*, *muffin*}
	3	05-01-2011:15.12	{*americano*, *cappuccino*}
	4	06-01-2011:11.18	{*espresso*, *muffin*}
	5	07-01-2011:14.24	{*cappuccino*}
	6	07-01-2011:14.24	{*americano*, *cappuccino*}
Mary	1	30-12-2010:11.32	{*tea*}
	2	30-12-2010:12.12	{*cappuccino*}
	3	30-12-2010:14.16	{*espresso*, *muffin*}
	4	05-01-2011:11.22	{*bagel*, *tea*}
Bill	1	30-12-2010:14.32	{*cappuccino*}
	2	30-12-2010:15.06	{*cappuccino*}
	3	30-12-2010:16.34	{*bagel*, *espresso*, *muffin*}
	4	06-01-2011:09.18	{*ristretto*}
	5	06-01-2011:12.18	{*cappuccino*}
...	...	...	...

Cada línea de la tabla corresponde a un cliente ordenando un conjunto de ítems. Hay una secuencia de órdenes por cliente. Las órdenes tienen una secuencia de números, un timestamp, y un conjunto de datos. El objetivo es encontrar una secuencia frecuente definida por un patrón como {cappuccino}, {espresso, muffin}, {cappuccino} para el primer cliente. Una secuencia es frecuente si contiene el patrón en una proporción predefinida de las secuencias del cliente en el conjunto de datos. Una secuencia *a1, a2, . . . , an* es una subsecuencia de otra *b1, b2, . . . , bm* si existen enteros $i1 < i2 < \cdots < in$ tal que $a1 \subseteq bi1$, $a2 \subseteq bi2$, . . . , $an \subseteq bin$. Por ejemplo la secuencia *{x}, {x,y}, {y}* es una subsecuencia de *{z}, {x}, {z}, {x, y, z}, {y, z}, {z}* porque $\{x\} \subseteq \{x\}$, $\{x,y\} \subseteq \{x, y, z\}$, y $\{y\} \subseteq \{y, z\}$. Sin embargo, *{x}, {y}* no es una subsecuencia de *{x,y}* y viceversa. El soporte de una secuencia *s* es la fracción de secuencias en el conjunto de datos que tienen s como subsecuencia. Una secuencia es frecuente si el soporte llega al umbral de *minsup*. En principio hay un número infinito de patrones potenciales. Sin embargo, igual que en el algoritmo Apriori, la propiedad monotonicidad se puede explotar: si una secuencia es frecuente, entonces sus subsecuencias también son frecuentes. Las secuencias frecuentes también se pueden utilizar para crear reglas de la forma $X \Rightarrow Y$ donde *X* es un patrón e *Y* es una extensión o continuación del patrón. Por ejemplo si *X= {cappuccino}, {espresso}* e *Y= {cappuccino}, {espresso}, {latte, muffin}*. Suponiendo que *X* tiene un soporte de 0.05 e *Y* tiene un soporte de 0.04, entonces la confidencia *de* $X \Rightarrow Y$ es *0.04/0.05 = 0.8,* es decir, 80% de los clientes que piden un cappuccino y que luego piden un espresso también piden un muffin y un latte. [4]

Minería de episodio

Otro problema que se puede resolver utilizando el enfoque tipo Apriori es el

descubrimiento de episodios frecuentes. Para este se utiliza una ventana móvil que analiza cuán frecuente ocurre un episodio. Un episodio define un orden parcial. El objetivo es descubrir episodios frecuentes.
La entrada para la minería de episodio es una secuencia de tiempo como se muestra en la figura 52

Figura 52 Una secuencia de tiempo de eventos y la ventana de tiempo correspondiente [4]

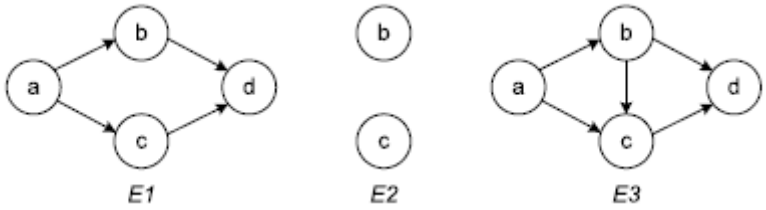


Figura 53 Tres episodios [4]

El tiempo de secuencia comienza a en el tiempo 10 y termina en el tiempo 37. La secuencia consiste en puntos de tiempo discretos, y como se ve en la figura en algunos puntos de tiempo ocurre un evento. Un evento tiene un tipo (ej. la actividad que ocurrió) y el timestamp.
Por ejemplo un evento con tipo *a* ocurre en el tiempo 12, un evento con tipo *c* ocurre en el tiempo 13, etc. En la figura 52 también se muestran 32 ventanas de tiempo de longitud 5. Estas son todas las ventanas que se superponen con la secuencia temporizada. La longitud 5 es un parámetro predefinido del algoritmo que se usa para descubrir patrones de secuencia. Un episodio ocurre en una ventana de tiempo si el orden parcial se embebe en él. La figura 53 ilustra tres episodios. Un episodio que se describe por un grafo acíclico dirigido. Los nodos se refieren a tipos de eventos y los arcos definen un orden parcial. Por ejemplo, el episodio *E1* define que *a* debe ser seguida por *b* y *c*, *b* debe ser seguido por *d*, y que *c* debe ser seguido por *d*. El episodio *E2* solo dice que *b* y *c* deben ocurrir al menos una vez. El episodio *E3* dice que *a* debe ser seguido por *b* y *c* , *b* debe ser seguido por *c,* b debe ser seguido por *d, y* c debe ser seguido por *d*. Este episodio tiene dos arcos redundantes: el arco de *a* a *c* y el arco de *b* a *d* se pueden remover sin cambiar los requerimientos. Un episodio ocurre en una ventana de tiempo si es posible asignar eventos a nodos en el episodio de manera que las relaciones de orden se satisfagan. Notar que el episodio solo define el conjunto mínimo de eventos, esto es, puede haber todo tipo de eventos adicionales. El requerimiento principal es que el episodio sea embebido. Para ilustrar la noción de "ocurrencia en una ventana de tiempo" utilizaremos a Figura 54.

Considerando el episodio *E1* y deslice la ventana de longitud 5 de izquierda a derecha. Hay 32 posiciones posibles. Sin embargo sólo una de las 32 ventanas comprende al episodio *E1* . Esta es la ventana comenzando en el tiempo 12. Aquí definimos la secuencia *a, c, b, e, d*. Claramente todos los requerimientos se encuentran en la secuencia: *a* se sigue por *c*, *b* es seguida por *d,* etc.

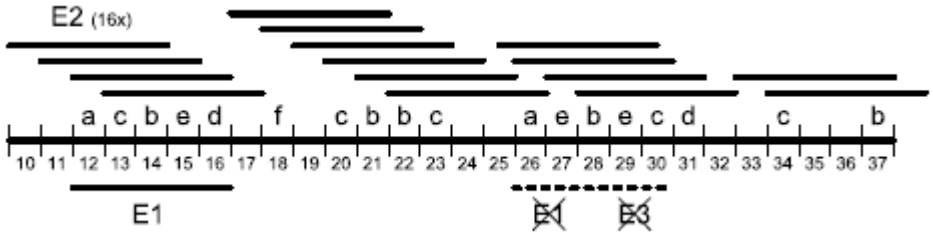


Figura 54: Ocurrencias del episodio *E1* y *E2 [4]*

El episodio *E2* con una ventana de longitud 5 ocurre 16 veces ya que solo hace falta que *c* y *b* ocurran en la misma ventana sin necesidad de una relación de orden.
El episodio *E3* no ocurre en la secuencia de tiempo si se utiliza una ventana de longitud 5. No hay ventana de longitud 5 donde la secuencia sea *a, b, c, d.* Si la longitud de la ventana se extiende a 6 entonces *E3* ocurre una sola vez.
El soporte de un episodio es la fracción de ventanas en el que ese episodio ocurre. Para una ventana con tamaño 5, el soporte para *E1* es 1/32 el soporte para *E2* es 16/32=0.5, y el soporte para *E3* es 0/32=0. Como en minería de secuencia y en aprendizaje de reglas de asociación, definimos un límite para el soporte. Todos los episodios que tienen un soporte de al menos ese límite son frecuentes. Por ejemplo si el límite es 0.2 entonces *E2* es frecuente mientras que *E1 y E3* no lo son.
El objetivo es generar todos los episodios frecuentes. Notar que por lo general hay muchos candidatos posibles (todas ordenes parciales sobre el conjunto de tipos de eventos). Afortunadamente, como en el algoritmo Apriori, podemos explorar la propiedad de monotonicidad para rápidamente descartar los candidatos no adecuados. Para explicar esta propiedad, debemos definir la noción de sub episodio. *E1* es un subepisodio de *E3* porque *E1* es un subgrafo de *E3,* es decir los nodos y arcos de *E1* están contenidos en *E3.* Es fácil ver que si un episodio *E* es frecuente, entonces todos los subepisodios son frecuentes. Esta propiedad de monotonicidad se puede usar para hacer speed-up en el proceso de búsqueda.
Los episodios frecuentes también se pueden usar para crear reglas de la forma $X \Rightarrow Y$ donde *X* es un subepisodio de *Y*. Como antes la confidencia de tal regla puede ser computada. En nuestro ejemplo, la regla $E1 \Rightarrow E3$ tiene una confidencia de 0/1=0 es decir muy pobre. La regla $E2 \Rightarrow E1$ tiene una confidencia de 1/16.
La minería de episodio y la minería de secuencia se pueden ver como variantes del aprendizaje de reglas de asociación, Por que toman en cuenta el orden de los eventos, y se encuentran relacionadas con el descubrimiento de procesos. Sin embargo hay muchas diferencias con los algoritmos de *Process Mining*. Primero y principal, solo patrones locales se consideran, es decir, no se genera el modelo de proceso en general. Segundo, se focaliza en comportamiento frecuente sin tratar de generar modelos que también excluyen comportamiento.
Los episodios no pueden modelar decisiones, loops. etc. Finalmente la minería de episodios

y la minería de secuencia no pueden manejar bien la concurrencia. La minería de secuencia sólo encuentra patrones secuenciales. La minería de episodio tiene problemas si suceden muchos episodios en concurrente, porque no queda claro que ventana de tiempo se debe seleccionar para obtener los episodios significativos.[4]

Otros enfoques

En las comunidades de data mining y el aprendizaje automático, muchas otras técnicas se han desarrollado para analizar la secuencia de eventos. Las aplicaciones se encuentran en minería de texto (secuencia de letras y palabras), bio-informaticas (análisis de secuencias de ADN), reconocimiento de discurso, web analytics, etc. Ejemplos de estas técnicas que se usan para este propósito son redes neuronales, y modelos de Markov escondidos. Las redes neuronales artificiales tratan de imitar el cerebro humano con el fin de aprender tareas complejas. Una red neuronal artificial, es un grupo interconectado de nodos, parecido a una red vasta de neuronas en el cerebro humano. Diferentes paradigmas de aprendizaje se pueden utilizar para entrenar una red neuronal: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje de reforzamiento. Las ventajas son que las redes neuronales pueden explotar el comportamiento paralelizado y que se pueden usar para resolver tareas mal definidas, imaginación y reconocimiento de voz. El mayor inconveniente es que el modelo resultante (una percepción multicapa), por lo general no es legible por el humano. Por lo que no hay modelo resultante.
Los modelos de Hidden Markov son una extensión de los procesos Markov ordinarios. Un modelo Markov hidden tiene un conjunto de probabilidades de estados y transiciones. Además, a diferencia de los modelos Markov estándar, en cada estado puede haber una observación, pero el estado por sí mismo permanece oculto. Las observaciones tienen probabilidades por estado como se muestra en la Figura 55.

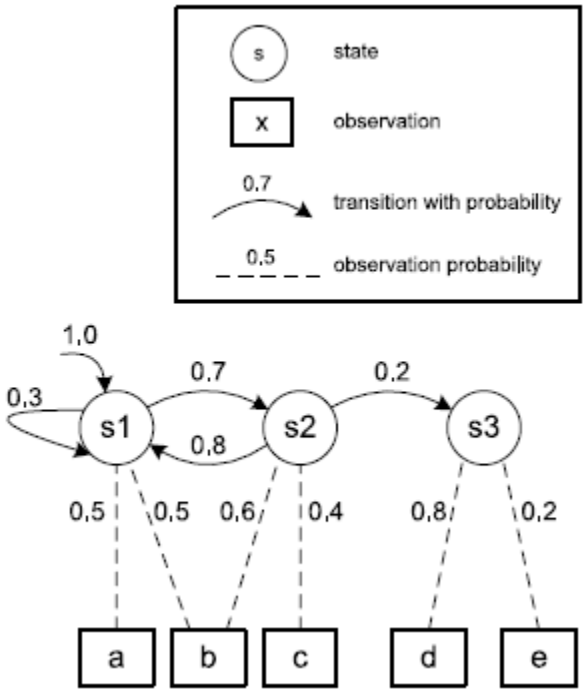


Figura 55 Un modelo de Markov hidden con tres estados: *s1*, *s2* y *s3*..[4]

Los arcos tienen probabilidades de transición de estados como se muestra, es decir, en el estado *s2* la probabilidad de cambiar al estado *s3* es 0.2 y la probabilidad de cambiar al estado *s1* es de 0.8. Cada visita a un estado genera una observación. Las probabilidades de observación son también dadas. Cuando se visita al estado *s2* la probabilidad de observación *b* es 0.6 y la probabilidad de observación de *c* es 0.4. Las posibles secuencias de observación son *{a,b,c,d,}, {a,b,b,c},* y *{a, b, c, b, b, a, c, e}*. Para la secuencia de observación *{a,b,c,d}*, esta claro que la secuencia escondida es *{s1,s2,s2,s3}*. Para las otras secuencias de observación, se pueden dar múltiples secuencias escondidas Hay Tres problemas fundamentales que se investigaron para modelos Markov hidden:

- Dada una secuencia de observación, como se computa la probabilidad de una secuencia dado un modelo de Markov hidden
- Dada una secuencia de observación y un modelo Markov hidden, cómo se computa el "camino más oculto" en el modelo.
- Dado un conjunto de secuencias observadas, como se deriva el modelo Markov hidden que maximiza la probabilidad de producir esas secuencias.

El último problema está más relacionado con *Process Mining*, pero también es el problema más difícil. El algoritmo Baum-Welch es también llamado algoritmo de Maximización de expectación que resuelve este problema para un número determinado de estados. A pesar de que los modelos de Markov hidden son versátiles y relevantes para *Process Mining*, hay varias complicaciones. Primero de todo hay muchos desafíos computacionales como el tiempo que consumen los procesos iterativos. Segundo, se debe adivinar un número apropiado de estados que sea entrada en el algoritmo. Tercero el modelo Markov resultante no es accesible por lo general para el usuario final, es decir, los modelos precisos son por lo general grandes y por lo general para ejemplos pequeños es difícil la interpretación del estado. [4]

A.1.2 Calidad de los modelos resultantes

Las técnicas de data mining enunciadas en el anexo 1 pueden ser explotadas para *Process Mining*, pero no se pueden usar directamente para las tareas más importantes de la misma como descubrimiento de procesos, chequeo de concordancia y mejora de procesos. Sin embargo hay otra razón para mostrar esta variedad de técnicas de data mining, al igual que en data mining, para *Process Mining* no es trivial analizar la calidad de los resultados. *Process Mining* se puede beneficiar de las experiencias en el campo de data mining. Mostraremos algunas de las técnicas de evaluación y validación desarrolladas para los algoritmos de data mining presentados.

A.1.2.1 Midiendo la performance de un clasificador

Ya se enunció previamente como construir un árbol de decisión. Hay muchas decisiones de diseño cuando se desarrolla un árbol de decisión (selección de atributos de separación, donde detener la separación, y determinar valores de corte). La pregunta es cómo evaluar la performance del aprendizaje de árboles de decisión. Esto es relevante para juzgar la integridad de los árboles de decisión resultantes y para comparar los diferentes enfoques. Una complicación es que uno solo puede juzgar la performance basándose en instancias vistas, aunque el objetivo también sea predecir buena clasificación para las instancias no vistas. Sin embargo, por simplicidad, primero asumiremos que queremos

juzgar el resultado de un clasificador (como los árboles de decisión) en un conjunto de datos dado.
Dado un conjunto de datos que consiste en *N* instancias, se sabe que para cada instancia cual es la clase actual y cuál es la clase predictora. Por ejemplo, para una persona particular que fuma, podemos predecir que la persona morirá joven (la clase predictiva es jóven) y sin embargo la persona muera a las 104 años(clase actual es viejo). Esto se puede visualizar con la matriz de confusión que se muestra en la figura 56. De los 200 estudiantes que fallaron, 178 clasificaron como que fallaron, 22 clasificaron como que aprobaron. Ninguno de los estudiantes que fallaron fueron clasificados con buen desepmeño (cum laude).De los 198 estudiantes que pasaron, 175 se clasificaron correctamente, 21 clasificaron como que fallaron y 2 con buen desempeño

		predicted class		
		failed	passed	cum laude
actual class	failed	178	22	0
	passed	21	175	2
	cum laude	1	3	18

Figura 56 Matriz de confusion para el árbol de decisión que se muestra en la Figura 49. [4]

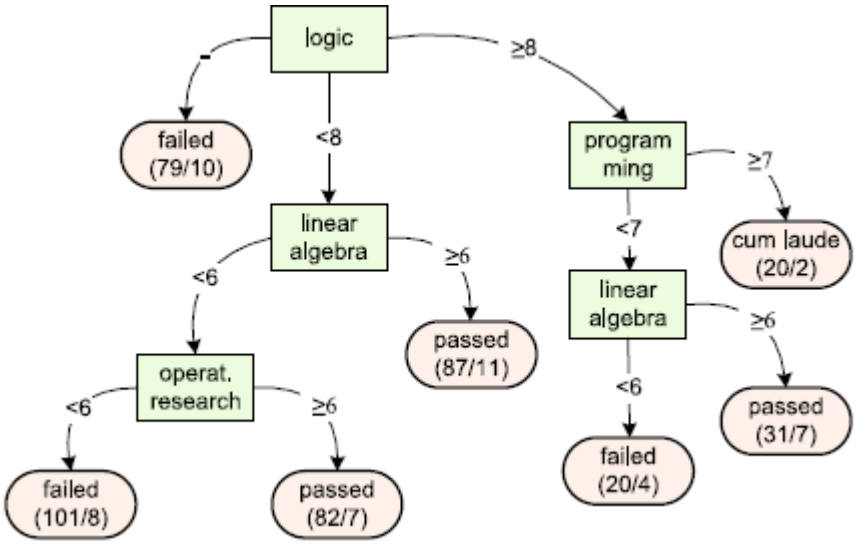


Figura 57 Un árbol de decisión, derivado de la tabla 1 [4]

Los 420 estudiantes se clasificaron en “fallo”, “paso” y “buen desempeño”, basándose en los resultados de los estudios. Los estudiantes que no tomaron el curso de lógica por lo general fallan (79 estudiantes de los cuales 10 no correspondieron a esta clasificación). Los estudiantes que tienen buenas notas en lógica y programación, por lo

general completan el curso con buen desempeño (20 estudiantes de los cuales 2 no correspondieron a esta clasificación)

El árbol de decisión de la figura 57 clasifica cada uno de los 420 estudiantes en clase actual y clase predictiva. Todos los elementos en la diagonal se predijeron correctamente, es decir 178 + 175 + 18 = 371 de los 420 estudiantes se clasificaron correctamente. Hay varias medidas de performance basadas en una matriz de confusión. Para definirlas consideraremos un conjunto de datos con solo dos clases "positivo" (+) y "negativo" (-) La figura 58 muestra la correspondiente matriz de confusión de 2x2. Las siguientes entradas se muestran:

- *tp* es el número de true positives (positivas verdaderas), es decir, instancias que son clasificadas correctamente como positivas.
- *fn* es el número de false negatives (negativas falsas), es decir, instancias que se predicen como negativas pero que deberían haber sido clasificadas como positivas.
- *fp* es el número de false positives (falsas positivas), es decir, instancias que se predicen como positivas pero deberían haber sido clasificadas como negativas.
- *tn* es el número de true negatives (negativas verdaderas), es decir, instancias que se clasifican correctamente como negativas.

En la figura 59(a) se muestra también la suma de las filas y columnas por ejemplo, $p = tp + fn$ es el número de instancias que son actualmente positivas, $n = fn + tn$ es el número de instancias que se clasifican como negativas por su clasificador. $N = tp + fn + fp + tn$ es el número total de instancias en el conjunto de datos. Basados en esto es fácil definir las medidas mostradas en la figura 59(b). El *error* se define como la proporción de instancias no clasificadas: $(fp + fn)/N$. La *precisión (accuracy)* mide la fracción de instancias en la diagonal de la matriz de confusión. El "true positive rate" (rango positivo verdadero), *tp-rate*, también conocido como "hit rate", mide la proporción de instancias positivas que han sido clasificadas como positivas. El "rango positivo falso", *fp-rate*, también llamado "rango de alarma falso", mide la proporción de instancias negativas que se clasifican incorrectamente como positivas. El término precisión y recuerdo provienen de la recopilación de información. La *precisión* se define como tp/p, se puede pensar a p como el número de documentos que han sido obtenidos basándose en una consulta de búsqueda y tp es el número de documentos que han sido obtenidos y que deberían haber sido obtenidos. El *recuerdo* (*recall*) se mide como tp/p donde p puede ser interpretada como el número de documentos que deberían haber sido obtenidos basándose en alguna consulta de búsqueda. Es posible tener alta *precisión* y bajo *recall*; pocos de los documentos buscados son devueltos en una consulta, pero esos que son devueltos son en realidad relevantes. Notar que el *recall* es lo mismo que el *tp-rate*. Hay otra frecuencia usada como métrica la llamada puntaje *F1*. Si la *precisión* o el *recall* es muy pobre (es decir cercano a 0), entonces el puntaje de *F1* también va a estar cercano a 0. Solo si la *precisión* y el *recall* son bueno entonces el puntaje de *F1* es cercano a 1.

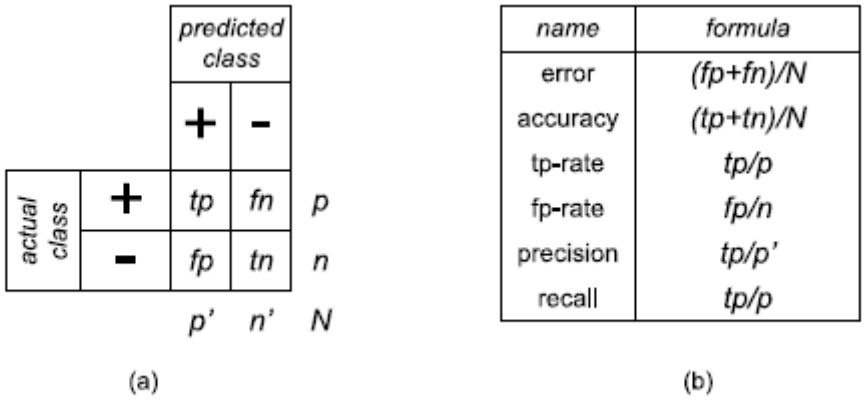


Figura 58 Matriz de confusión de dos clases y algunas medidas de performance para los clasificadores.[4]

Dos ejemplos de matriz de confusión del árbol de decisión extraído de los fumadores expuestos previamente son los siguientes:

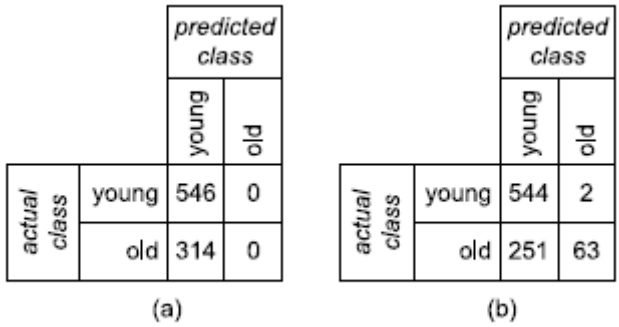


Figura 59 Dos matrices de confusión de los árboles de decisión [4]

En los dos primeros árboles de decisión, todas las instancias son clasificadas como jóvenes. Notar que incluso después de la separación del nodo raíz basándose en el atributo smoker, todavía todas las instancias se predicen que morirán antes de los 70. En la figura 59(a) se muestra la correspondiente matriz de confusión *asumiendo "young = positive" y "old = negative". N = 860, tp = p = 546, y fp = n = 314.* Notar que n'=0 or que todas están clasificadas como jóvenes. El *error* es de (314 + 0)/860 = 0.365, el *tp-rate* es 546/546 = 1, el *fp-rate* es 314/314 = 1, la *precisión* es 546/860 = 0.635, *recall* es 546/546 = 1. La Figura 59 muestra la matriz de confusión para el tercer árbol de decisión. El *error* es de (251 + 2)/860 = 0.292, *el tp-rate* es

544/546 = 0.996, el *fp-rate* es 251/314 = 0.799, *precisión* es 544/795 = 0.684, *recall* es 544/546 = 0.996. Por lo tanto, como se esperaba, la clasificación mejora: el *error* y el *fp-rate* decrecen considerablemente y el *tp-rate, precision* y *F1* se incrementan. Notar que el *recall* bajo poco por las dos personas que ahora se predicen que viven pero que no (a pesar de no fumar y no beber). [4]

Validación ***cruzada***

Toda la variedad de métricas de performance computadas usando la matriz de confusión se basan en el mismo conjunto de datos. Por lo tanto la matriz de confusión solo nos está diciendo algo sobre las instancias vistas, es decir, instancias usadas para aprender clasificación. En general, es trivial proveer clasificadores que se adecuen perfectamente (es decir que *precisión*, *recall* y *F1* tomen todas 1) en las instancias vistas. (Aquí, asumimos que las instancias son únicas o son instancias con atributos idénticos que pertenecen a la misma clase). Por ejemplo si los estudiantes tuvieran un único número de registración, entonces el árbol de decisión puede tener un nodo hoja por cada estudiante representando perfectamente el conjunto de datos. Sin embargo, esto no dice nada sobre las instancias no vistas, es decir el número de registración de un nuevo estudiante no tiene información sobre la performance esperada de ese estudiante.
El criterio más obvio para estimar la performance de un clasificador es *la precisión predictiva* en las instancias no vistas. El número de instancias no vistas es potencialmente grande (si no es infinito), por lo tanto un estimativo se tiene que computar en un conjunto a testear.
Esto se identifica como *validación cruzada*. El conjunto de datos se separa en conjunto de entrenamiento y conjunto de testeo. El conjunto de entrenamiento se usa para aprender un modelo mientras que el conjunto de testeo se usa para evaluar el modelo basándose en ejemplos no vistos.
Es importante notar que la *validación cruzada* no se limita a clasificación, y se puede usar para cualquier técnica de data mining. El único requerimiento de la *validación cruzada* es que la performance del resultado se pueda medir de una sola forma. Para clasificación, definimos medidas como *precisión*, *recall*, *F1 score*, y *error*.

Para regresión, se pueden definir varias medidas. En el contexto de regresión lineal, el *mean square error* es un indicador estándar de calidad. Si *y1, y2, ..., yn* son los valores actuales y $\hat{y}1, \hat{y}2, \ldots, \hat{y}n$ son los valores predictivos acorde al modelo de regresión lineal entonces $(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2)/n$ es el error del cuadrado medio.

Clustering se usa por lo general de una forma más descriptiva, y raramente se usa para hacer predicciones directas sobre instancias no vistas. Sin embargo, los clusters derivados para un conjunto de entrenamiento se pueden testear sobre un conjunto de testeo. Asignar todas las instancias en el conjunto de testeo a el centroide más próximo. Luego de hacer eso, la distancia promedio de cada instancia a su centroide se puede usar como una medida de performance.

En el contexto de minería de reglas de asociación, definimos métricas como soporte, confidencia, y elevamiento. Se puede aprender reglas de asociación, usando un conjunto de entrenamiento y luego testear las reglas descubiertas con el conjunto de testeo. La métrica de confidencia entonces indicará la proporción de instancias para las que la regla es soportada mientras sea aplicable. Luego definiremos estas reglas para *Process Mining*. Por ejemplo dado un log de eventos que sirva como conjunto de testeo y una red de Petri como modelo, se puede ver la proporción de instancias que pueden ser reproducidas en el modelo.

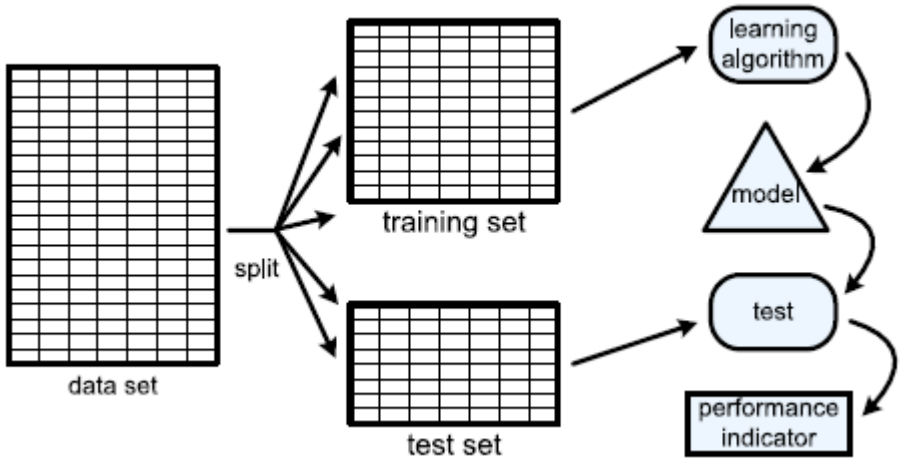


Figura 60 Validación cruzada utilizando un conjunto de testeo y otro de entrenamiento. [4]

La Figura 60 muestra la configuración básica de la validación cruzada. El conjunto de datos se separa en el conjunto de testeo y de entrenamiento. Basándose en el conjunto de entrenamiento, se genera un modelo (es decir un árbol de decisión o un modelo de regresión). Luego la performance se analiza usando el conjunto de testeo. Si solo se genera un número como indicador de performance, entonces no se dará una indicación de la confiabilidad del resultado. Por ejemplo, basándonos en un conjunto de testeo el puntaje de *F1* es 0.811. Sin embargo, basándose en otro conjunto de testeo el puntaje de *F1* puede ser un valor completamente distinto aunque las circunstancias no cambien. Por lo tanto, siempre se quiere calcular un intervalo confidente como indicador de performance. Los intervalos confidentes se pueden calcular sobre múltiples mediciones. Aquí discutiremos dos posibilidades:

- La primera posibilidad es la que mide un indicador de performance que es el promedio sobre un conjunto grande de mediciones independientes. Considerar por ejemplo la clasificación. El conjunto de testeo consiste en N instancias que son mutuamente independientes. Por lo tanto, cada clasificación $1 \leq i \leq N$ se puede ver como un test separado xi donde xi=1 quiere decir que la clasificación está mal y si $xi=0$ la clasificación es buena. Estos test se pueden ver como ejemplos de una distribución de Bernoulli con parámetro p (p es la probabilidad de una calificación mala). Esta distribución tiene un valor esperado p y una varianza de p (1 − p). Si asumimos que N es grande, entonces el error promedio $(\sum_{i=1}^{n} x_i)/N$ se distribuye normal aproximadamente. Esto es por el teorema central del límite, también conocido como "regla de números grandes". Asumiendo esto, encontramos que el intervalo de confidencia es 95% que es $[p - \alpha_{0.95}\sqrt{p(1-p)/N}, p + \alpha_{0.95}\sqrt{p(1-p)/N}]$, es decir, con certeza de 95% el error promedio real caerá dentro de $p - \alpha_{0.95}\sqrt{p(1-p)/N}$ and $p + \alpha_{0.95}\sqrt{p(1-p)/N}$. α0.95 = 1.96 es el valor estándar que se pueden encontrar en cualquier libro de estadística, p es la

tasa de error promedio medida, y *N* es el número de tests. Para calcular el intervalo confidente de 90% o el 99% se puede usar α0.90 = 1.64 y α0.99 = 2.58 respectivamente. Notar que sólo es posible calcular ese intervalo si hay muchas posibles mediciones independientes basadas en una solo ejecución del test.

- La segunda posibilidad es validación cruzada en k pliegos. Este enfoque se utiliza cuando hay pocas instancias en el conjunto de datos o cuando el indicador de performance se define en el conjunto de instancias en vez de en una sola instancia. Por ejemplo el puntaje *F1* no se puede definir para una instancia aislada. La figura 61 ilustra la idea de la validación cruzada en k pliegos. El conjunto de datos se separa en k partes iguales ej. K=10. Luego se realizan los *k* tests. En cada test, uno de los subconjuntos sirve como un conjunto de testeo mientras que los otros *k-1* subconjuntos sirven juntos como conjunto de entrenamiento. Si el subset $i \in \{1, 2, \ldots, k\}$ se usa como conjunto de test, luego la unión de los subconjuntos $\{1, 2, \ldots, i-1, i+1, \ldots, k\}$ se usa como conjunto de entrenamiento. Se pueden inspeccionar los tests individuales o tomar el conjunto de los k pliegos.

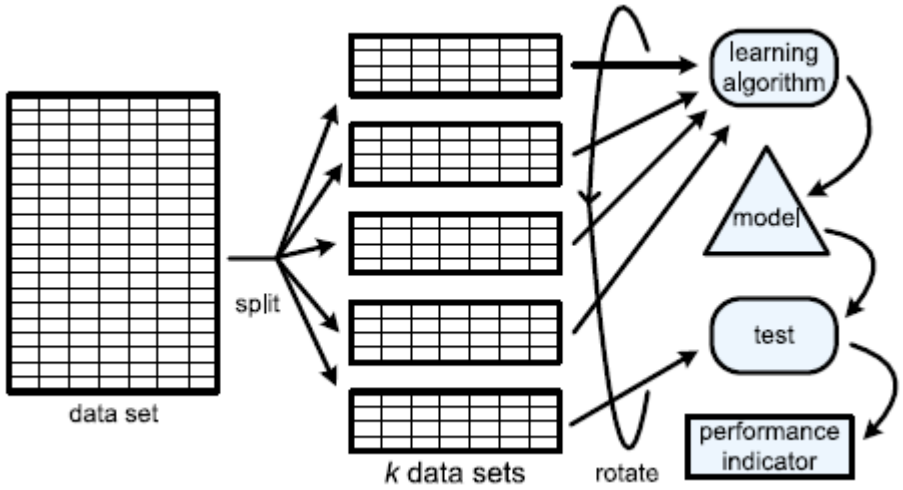


Figura 61 Validación cruzada de k-pliegos [4]

Hay dos ventajas asociadas a la validación cruzada en k pliegos. Primero de todo, todos los datos se usan tanto como datos de entrenamiento y datos de test. Segundo, si se desea, se pueden tener k tests del indicador de performance deseado en vez de uno solo. Formalmente los test no se pueden considerar como independientes ya que los conjuntos de entrenamiento usados en los k pliegos se superponen considerablemente. Sin embargo, las k pliegues hacen posibles tener más profundización en la fiabilidad.
Una variante extrema de la validación cruzada en k-pliegos es la validación cruzada de "dejar uno afuera" también conocida como jack.knifing. Aquí *K=N*, es decir el conjunto de test contiene sólo un elemento en ese momento. [4]

Navaja de Occam

Evaluar la calidad de los resultados de data mining no está nada cerca de ser trivial. Discutiremos en esta sección algunas complicaciones adicionales que son también

relevantes para *Process Mining*.

El aprendizaje es por lo general un problema mal planteado, es decir solo se dan ejemplos. Algunos ejemplos pueden excluir ciertas soluciones, sin embargo, muchos modelos permanecen posibles. Además, por lo general hay parcialidad en la presentación de objetivos y el algoritmo de aprendizaje. Considerar, por ejemplo, la secuencia 2, 3, 5, 7, 11, ... Cuál es el próximo elemento en la secuencia? La mayoría dirá que es 13, es decir, el próximo número primo, pero hay muchas secuencias que comienzan con 2, 3, 5, 7, 11. Por lo que, hay una preferencia para la formulación de hipótesis de algunas soluciones. El término *sesgo inductivo* se refiere a la preferencia de una solución ante otra que no se puede determinar por el conjunto de datos por sí mismos pero que se maneja por factores externos.

Un *sesgo representacional* se refiere a las decisiones que se toman implícitamente mediante la selección de una representación particular. Por ejemplo en la de árboles de decisión asumimos que el mismo atributo puede aparecer solo una vez en un camino. Este *sesgo representacional* obtiene ciertas soluciones, por ejemplo, un árbol de decisión donde cerca a la raíz se usa un atributo numérico en forma de grano grueso y en algunos subárboles se usa en forma de grano fino. La regresión lineal hace supuestos sobre la función usada para adecuarse mejor a los datos. La función se asume lineal aunque pueda haber funciones no lineales que se adecuen mejor. Notar que un sesgo representacional no necesariamente es malo, por ejemplo, la regresión lineal se ha utilizado exitosamente en varios dominios de aplicación. Sin embargo, es importante darse cuenta que el espacio buscado se limita por la representación usada. Las limitaciones pueden guiar el proceso de búsqueda, pero también excluir soluciones buenas.

Un *sesgo de aprendizaje* referencia a estrategias usadas por el algoritmo que da preferencias a soluciones particulares. En la figura 62 usamos el criterio de información ganada (reducción de entropía) para seleccionar los atributos. Sin embargo, podríamos haber usado el índice de diversidad de Gini *G* en vez de la entropía *E* para seleccionar los atributos, por lo tanto obteniendo diferentes árboles de decisión resultantes.

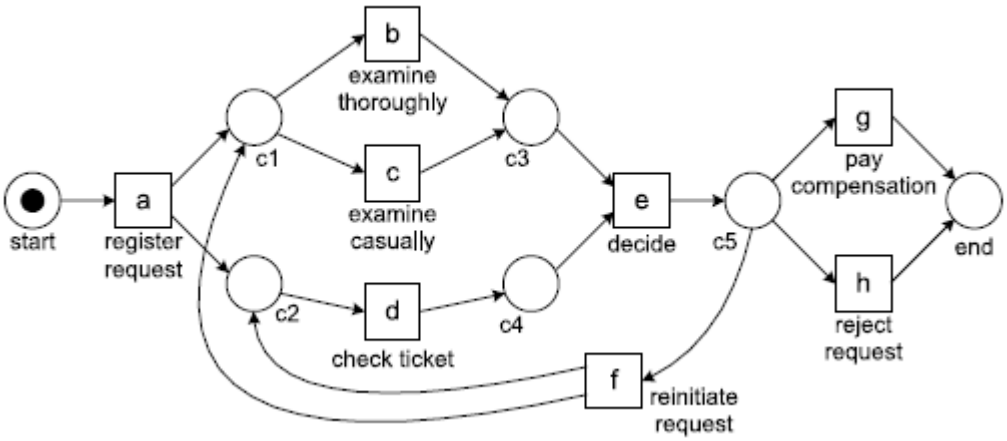


Figura 62 modelo de proceso descubierto utilizando el algoritmo α basándose en las siguientes instancias de proceso *{(a, b, d, e,h), (a, d, c, e, g), (a, c, d, e, f, b, d, e, g), (a, d, b, e,h), (a, c, d, e, f, d, c, e, f, b, d, e,h), (a, c, d, e, g)}.[4]*

Ambos factores también juegan un rol en *Process Mining*. Considere por ejemplo

la figura 62. Este modelo de proceso se descubrió utilizando el algoritmo α basándose en las siguientes instancias de proceso *{(a, b, d, e,h), (a, d, c, e, g), (a, c, d, e, f, b, d, e, g), (a, d, b, e,h), (a, c, d, e, f, d, c, e, f, b, d, e,h), (a, c, d, e, g)}*. Claramente hay un sesgo representacional.

El supuesto es que el proceso se puede representar con una red de Petri donde toda transición lleva una etiqueta visible y única. Muchos procesos no pueden ser representados con una red de Petri. El algoritmo α tiene también un sesgo de aprendizaje ya que se focaliza en sucesiones directas. Si *a* es seguida directamente por *b,* entonces esta información es utilizada. Sin embargo una observación como *"b* eventualmente sigue a *a* en el log de eventos" no es explotada por el algoritmo α.

Un sesgo inductivo no es necesariamente malo. En realidad a veces se requiere para obtener una solución. Sin embargo el analista debe ser consciente de esto y reflejarlo en las decisiones implícitas tomadas.

Otro problema es el balance delicado entre el *overfitting* y el *underfitting*. Un modelo aprendido tiene overfitting si es muy específico y tomó en cuenta información accidental en el conjunto de datos. Por ejemplo cuando se construye un árbol de decisión de un conjunto de entrenamiento sin entrada conflictiva (es decir, instancias con atributos idénticos dentro de la misma clase), es fácil construir el árbol de decisión con un puntaje *F1* perfecto. Este árbol se puede obtener mediante la continua separación de nodos hasta que cada nodo hoja corresponda a instancias pertenecientes a la misma clase. Sin embargo, es obvio que semejante árbol de decisión es muy específico y tiene poco valor predictivo.

Un modelo aprendido tiene *underfitting* si es muy general y permite cosas que no son soportadas por evidencia en el conjunto de datos. Mientras que el *overfitting* se puede caracterizar por la falta de generalización, el *underfitting* tiene el problema opuesto: mucha generalización. Considerar por ejemplo la generación de reglas de asociación. Generando muchas reglas detalladas debido a una configuración muy baja de *minsup* y *minconf*, corresponde a overfitting. Muchas reglas se encuentran, pero estas son probablemente más específicas para el conjunto de entrenamiento. Generar muy pocas reglas debido a una alta configuración de *minsup* y *minconf*, corresponde a *underfitting*. En el caso más extremo no se obtiene ninguna regla de asociación. Notar que un modelo sin reglas se adecúa a cualquier conjunto de datos y, por lo tanto, no contiene información.

El underfitting es problemático si el conjunto de datos no contiene ejemplos negativos. Suponer que tenemos un conjunto de entrenamiento con valores positivos únicamente, es decir, n=0 en el conjunto de entrenamiento. Como construir un árbol de decisión sin ejemplos negativos? La mayoría de los algoritmos clasificarían todo como positivo. Esto muestra que la clasificación asume tanto ejemplos positivos como negativos. Este no es el caso del aprendizaje de reglas de asociación. Si consideramos al conjunto de datos de la tabla 2 y suponemos que el conjunto de ítems *{latte, tea, bagel}* no aparece en el conjunto de datos. Esto quiere decir que ningún cliente ordenó estos tres ítems juntos en un conjunto de entrenamiento. Podemos concluir de esto que no es posible ordenar estos tres ítems juntos? Por supuesto que esto es incorrecto. Por lo tanto, el aprendizaje de reglas de asociación se focaliza en ejemplos positivos que son por lo general frecuentes. Sin embargo, para algunas aplicaciones sería útil poder descubrir "reglas negativas" como las regla de que el cliente no pueden pedir latte, tea y bagels en una misma orden.

Un buen balance entre el *overfitting* y el *underfitting* es de lo más importante del descubrimiento de procesos. Considere por ejemplo la red de Petri de la figura 62. El modelo permite el comportamiento observado en el log de eventos. También está

generalizado y permite más secuencias de las presentadas en el conjunto de entrenamiento. Es decir que el modelo tiene *underfitting*. Este dilema es causa de la falta de ejemplos negativos en el log de eventos. Las instancias en el log de eventos muestran lo que ha sucedido pero no muestran lo que no puede suceder.

La navaja de Occams es un principio que enuncia lo siguiente: "uno no debe incrementar, más allá de lo necesario, el número de entidades requeridas para explicar algo", es decir, uno debe buscar el modelo más simple que pueda representar lo que se observa en el conjunto de datos. Éste principio está relacionado con encontrar el balance natural entre *overfitting* y *underfitting*. El principio de descripción de longitud (*MDL*) trata de operacionalizar la navaja de Occam. De acorde a este paradigma, la calidad del modelo no se basa solamente en la performance predictiva (ejemplo el puntaje de *F1*), sino que también en la simplicidad del modelo. Además no es el objetivo de la validación cruzada. En *MDL*, la performance se juzga con el conjunto de entrenamiento solo y no medido contra las instancias no vistas nuevas. Esta idea básica es que el mejor modelo es aquel que minimice la codificación tanto del modelo como del conjunto de datos. Aquí se usa la profundización de que cualquier irregularidad en el conjunto de datos se puede usar para comprimir los datos, es decir, para describir que se utilizan menos símbolos que el número de símbolos necesarios para describir los datos literalmente. Cuantas más regularidades haya, se pueden comprimir más los datos. Igualando "aprendizaje" con "encontrar regularidades", implica que cuanto más posibilidades de comprimir los datos, más hemos aprendido del conjunto de datos. Obviamente, el conjunto de datos se puede codificar más compactamente si se captura del modelo conocimiento valuable del conjunto de datos. Sin embargo, codificar ese conocimiento también requiere espacio. Un modelo complejo y con *overfitting* ayuda a reducir el codificado del conjunto de datos. Un modelo simple y con *underfitting* se puede almacenar más compactamente, pero no ayuda a reducir el codificado del conjunto de datos. Esta idea está relacionada con la noción de entropía en el aprendizaje de árboles de decisión. Cuando se construía el árbol de decisión, el objetivo es encontrar nodos hoja homogéneos que se puedan codificar compactamente. Sin embargo, no mencionamos una penalización por la complejidad del árbol de decisión por sí mismo. El objetivo de *MDL* es minimizar la entropía del conjunto de datos codificado utilizando el modelo aprendido, y el codificado del modelo en sí mismo. Para medir el balance entre el *overfitting* y el *underfitting*, los pesos de las variables se pueden asociar a las dos codificaciones.

Aplicar la navaja de Occams no es fácil. Extraer puntos de vista confiables y significativos de datos complejos está lejos de ser trivial. De hecho, es mucho más fácil transformar un conjunto complejo de datos en "basura impresionante" abusando de las técnicas presentadas. Sin embargo, utilizando sabiamente a data mining se puede agregar mucho valor.
Además, *Process Mining* agrega la "dimensión de proceso" a los datos y se puede usar para analizar datos de eventos desde una perspectiva más holística. Como mostraré más adelante *Process Mining* crea un puente sólido entre el modelado de procesos y análisis por un lado y data mining por otro. [4]

Anexo 2 Formato Log de eventos

En este anexo se ve el detalle la información necesaria para un log de eventos y se profundiza en el estándar XES.

A.2.1 Log de Eventos

Un log de eventos debe contener información relacionada con un solo proceso. El primer alcance de grano grueso debe garantizar eso. Además cada evento en el log se debe referir a una sola instancia de proceso, también referida como caso. Los eventos pueden también referenciar a una actividad. El identificador de caso y la actividad son la base del log de eventos. Los eventos dentro de un caso deben estar ordenados. Sin el orden de la información es imposible descubrir dependencias causales en modelos de procesos. En la Figura 63 se muestra la estructura de árbol de un log de eventos.

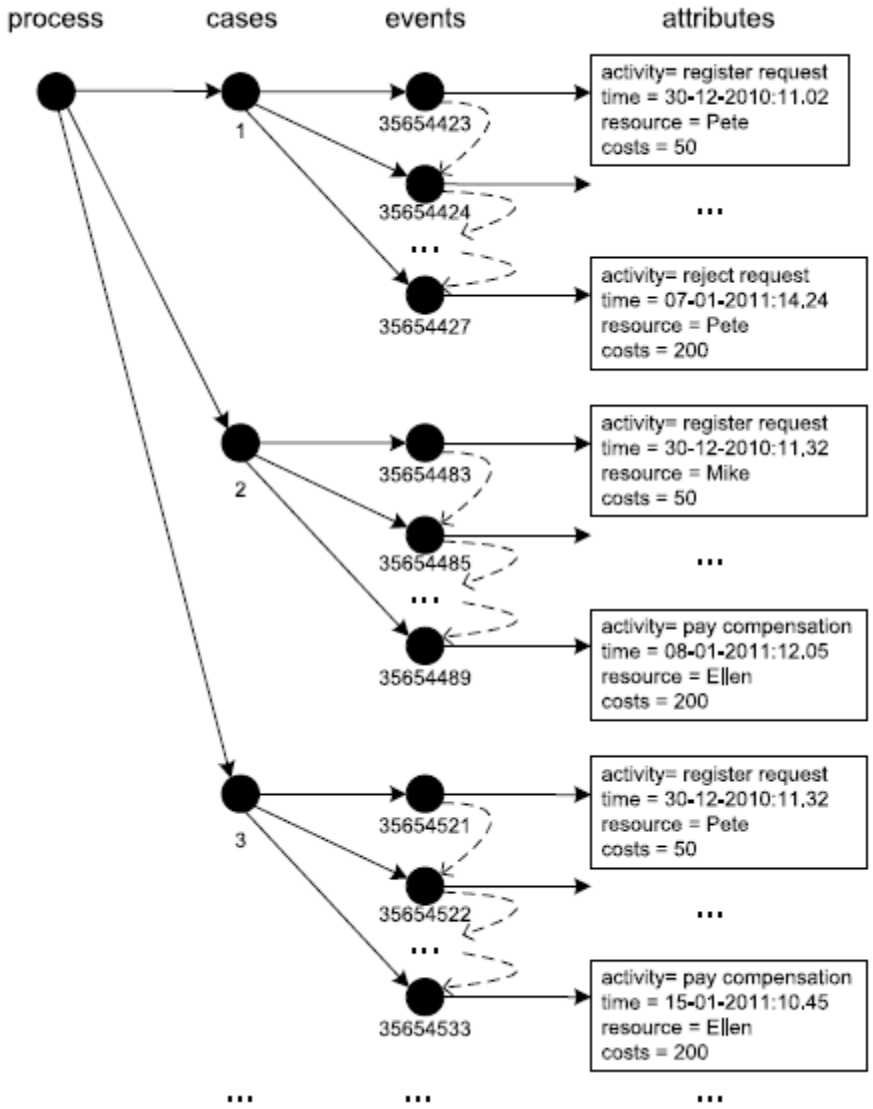


Figura 63 Estructura de un log de eventos [4]

Suposiciones de log de eventos:

- Un proceso se compone de casos
- un caso consiste en eventos, por lo que cada evento se relaciona precisamente con un caso.
- Los eventos en un caso están ordenados.
- Los eventos pueden tener atributos. ejemplos de típicos atributos son actividades, tiempo, costos recursos.

No todos los eventos necesitan tener el mismo conjunto de atributos, sin embargo por lo general eventos que se refieren a la misma actividad tienen los mismos atributos.

Definición: (Evento, atributo) Sea E el universo de eventos, es decir, el conjunto de todos los identificadores de eventos posibles. Los eventos se pueden caracterizar por varios atributos, es decir, un evento puede tener un timestamp, puede corresponder a una actividad, se puede ejecutar por una persona en particular, tener costos asociados, etc. Sea AN el conjunto de nombres de atributos. Para cada $e \in E$ y cada nombre $n \in AN$: $\#n(e)$ es el valor del atributo n para un evento e. Si el evento e no tiene un atributo llamado n entonces $\#n(e)=\perp$ (valor nulo).

Por conveniencia asumimos los siguientes atributos estándar:

- *#activity(e)* es la actividad asociada al evento *e*.
- *#time(e)* es el timestamp del evento *e*.
- •*#resource(e)*es el recurso asociado al evento *e*.
- *#trans(e)* es el tipo transaccional asociado al evento e, como por ejemplo empezado, completado, suspendido.

Puede suceder que en el log de eventos haya eventos que sean de la misma actividad para el mismo caso ya sea como en los loops, o ejecución de actividades en paralelo. Esto se llama problema de correlación secundario esto es relacionar dos eventos con el mismo caso. Este problema puede solucionarse mediante la adición de información al log de eventos o utilizando heurística.
Las técnicas de *Process Mining* pueden utilizarse para descubrir automáticamente modelos de procesos. En estos modelos de procesos las actividades juegan un papel central. Puede suceder que muchos eventos correspondan a la misma actividad diferenciados por el atributo transaccional, es decir una tarea puede estar empezada, completada, suspendida, etc. Algunas técnicas de *Process Mining* tienen en cuenta el atributo transaccional mientras que otras no y las consideran como eventos atómicos. Esto se puede realizar mediante el filtro (por ejemplo removiendo eventos de un tipo particular) y por el concepto de clasificador.

Un clasificador es una función que mapea atributos de un evento en una etiqueta que se utiliza en el modelo de procesos resultante. Esta puede ser vista como el nombre del evento. En un principio puede haber muchos clasificadores pero se utiliza uno a la vez.

Definición (Clasificador) para cada evento e ∈ E , e es el nombre del evento.

Si los eventos se definen por el nombre de la actividad y el tipo transaccional, entonces $e = (\#activity(e), \#trans(e))$. Ahora la instancia de la actividad a ser mapeara en *(a, schedule), (a, assign), (a, start), (a, complete)* y el α-algorithm básico creara cuatro transiciones refiriéndose al ciclo de vida de a.

Un log de eventos consiste en casos y los casos consisten en eventos. Los eventos para un caso se representan en la forma de trazo, esto es una secuencia de eventos únicos. Por otra parte los casos, así como los eventos, pueden tener atributos.

Definición. (Caso, trazo, log de eventos) Sea C el universo de casos, es decir el conjunto de todos los identificadores de casos posibles. Los casos como los eventos, tienen atributos. Para cualquier caso $c \in C$

Y con nombre $n \in AN: \#n(c)$ es el valor del atributo n para el caso c ($\#n(c)=\perp$.si el caso c no tiene un atributo llamado n). Cada caso tiene un atributo de tace obligatorio: $\#trace(c) \in E*.1$ ^ c = #trace(c) es la forma corta de referenciar al trace de un caso.

Un trace es una secuencia de eventos finita $\sigma \in E*$ de manera que cada evento aparece una única vez, es decir para $1 \leq i < j \leq |\sigma|: \sigma(i) = \sigma(j)$.

Un log de eventos es un conjunto de casos $L \subseteq C$ tal que cada evento aparece como mucho una vez en el log de eventos entero, es decir para cada $c1, c2 \in L$ tal *que* $c1 = c2: \partial set(c\hat{}1) \cap \partial set(c\hat{}2) = \emptyset$.

Si el log de eventos contiene timestaps el orden en un trace debe respetar esos timestamps.

Los eventos y los casos deben tener identificadores únicos, este mecanismo nos permite referirnos a un evento especifico o a un caso específico. Esto puede ser importante porque puede haber muchos eventos que tengan los mismos atributos. Puede haber distintos casos que sigan el mismo camino en el proceso. Estos son identificadores técnicos que nos ayudan a señalar a un evento o caso particular. Por esto no es necesario que existan en la fuente de datos original y se pueden generar cuando se extraen los datos de las distintas fuentes de datos.

Los eventos y los casos pueden tener cualquier número de atributos, utilizando el mecanismo de clasificación cada evento obtiene un nombre. Por lo tanto a veces requerimos que los eventos tengan un atributo de actividad. Los casos tienen un atributo camino que indica la secuencia de eventos que fueron llevados a cabo por el caso c. $\hat{}c = \#trace(c)$.

Mediante la formalización de log de eventos de esta forma, formulamos precisamente los requerimientos que imponemos en el log de eventos sin discutir una sintaxis concreta.

Es más podemos utilizar la representación formal para consultar el log de eventos y utilizarlo como punto de partida para análisis:

- $\{\#activity(e) \mid c \in L \wedge e \in \hat{}c\}$ es el conjunto de todas las actividades q aparecen en el log L.
- $\{@resource(e) \mid c \in L \wedge e \in \hat{}c \wedge \#trans(e)=manualSkip\}$ es el conjunto de los recursos que se han salteado una actividad.
- $\{a \in A \mid c \in L \wedge a = \#activity(c\hat{}(1)) \wedge a= \#activity\ (c\hat{}(|c\hat{}|))\}$ es el conjunto de todas las actividades que sirvieron como actividades de inicio y fin para el mismo caso.

Definición (Log de eventos simple) Sea A el conjunto de los nombres de las actividades. Un trace simple σ es una secuencia de actividades, es decir, $\sigma \in A*$. Un log de eventos simple L es un conjunto múltiple de traces sobre A, es decir, $L \in B(A*)$. Un log de eventos simples es solo un conjunto múltiple de caminos sobre A (el conjunto de actividades). Por ejemplo *[(a, b, c, d)3, (a, c, b, d)2, (a, e, d)]* define un log q contiene 6 casos. En total hay 23 eventos. Todos los casos empiezan con *a* y termina en *d*. En un log no siempre hay atributos por ejemplo los timestamp y los recursos son extraídos. Además los casos y los eventos no tienen identificadores únicos. [4][3]

A.2.1.1 Estándar XES

Hasta hace poco el estándar por defecto para almacenar e intercambiar logs de eventos era el MXML (mining eXtensible Markup language). MXML emergió en el 2003 y luego se adoptó por ProM. Utilizando MXML se puede almacenar un log de eventos utilizando una sintaxis basada en XML.
ProMimport es una herramienta que soporta la conversión de diferentes fuentes de datos a MXML, como por ejemplo MS Access, Aris PPM, CSV, Apache, PeopleSoft, Subversion, SAP R/3. Protos, Cognos. MXML tienen una notación estándar para almacenar timestamps, recursos y tipos de transacciones. Además se pueden agregar arbitrariamente atributos a eventos y casos. El resultado posterior es un MXML ad-hoc con ciertos atributos de datos que se interpretan de una forma específica.
XES es el sucesor de MXML. Se basa en muchas experiencias prácticas que se tuvieron con MXML, el formato XES se hizo menos restrictivo y realmente extensible. Es el formato adoptado por la IEEE Task Force on *Process Mining*. El formato es adoptado por herramientas como ProM, Nitro, XESme, y OpenXES.

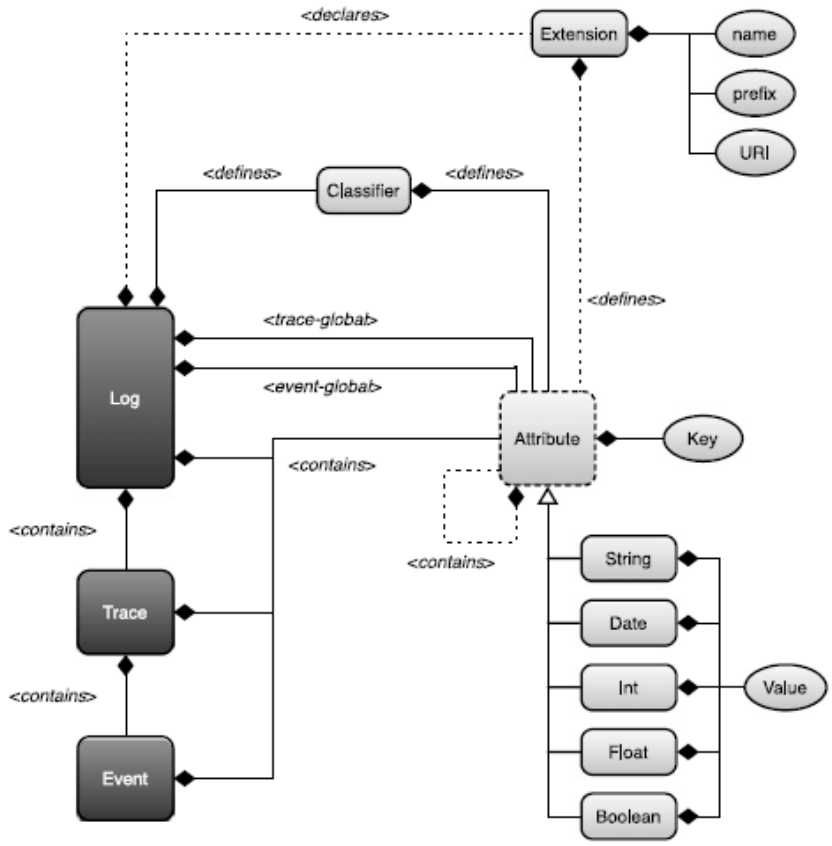


Figura 64 Modelo Meta de XES.[4]

En la Figura 64 se muestra el modelo meta de XES como se ve un documento XES (es decir un archivo XML) contiene un log que consiste de un número de rastros. Cada rastro describe una lista secuencial de eventos correspondientes a un caso particular. El log, sus rastros, y sus eventos pueden tener un número de atributos . Los atributos pueden estar anidados. Hay cuatro tipos principales: String, Date, Int, Float y Boolean. Estos corresponden al estándar de tipos XML xs:string, xs:dateTime, xs:long, xs:double, and xs:boolean. Las extensiones pueden definir nuevos atributos y un log debe declarar todas las extensiones utilizadas en el mismo. Los atributos globales son atributos que se declaran para que sean obligatorios. Tales atributos residen a nivel de trace o de evento. Los atributos pueden ser anidados. Los clasificadores de eventos se definen para el log y asignan una etiqueta (por ejemplo el nombre de la actividad) a cada evento. Puede haber múltiples clasificadores.
XES no impone número de atributos para cada elemento (log, trazo, o evento). Para proveer una semántica a los atributos el log se refiere a estos como extensiones. Una extensión da semántica a un atributo particular. Por ejemplo la extensión tiempo define al atributo timestamp. La extensión organizacional corresponde al atributo recurso de tipo String. En XES no se necesitan identificadores únicos para los casos y los eventos. De hecho, se puede pensar en la posición en el log como el identificador del evento o del caso. XES puede declarar que atributos particulares sean obligatorios. Por ejemplo se puede setear que cada rastro tenga un nombre o que cada evento tenga un timestamp. Para este

propósito un log contiene dos listas de atributos globales: una para los rastros, y otra para los eventos.

Table 4.1 A fragment of some event log: each line corresponds to an event

Case id	Event id	Properties				
		Timestamp	Activity	Resource	Cost	...
1	35654423	30-12-2010:11.02	Register request	Pete	50	...
	35654424	31-12-2010:10.06	Examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	Check ticket	Mike	100	...
	35654426	06-01-2011:11.18	Decide	Sara	200	...
	35654427	07-01-2011:14.24	Reject request	Pete	200	...
2	35654483	30-12-2010:11.32	Register request	Mike	50	...
	35654485	30-12-2010:12.12	Check ticket	Mike	100	...
	35654487	30-12-2010:14.16	Examine casually	Pete	400	...
	35654488	05-01-2011:11.22	Decide	Sara	200	...
	35654489	08-01-2011:12.05	Pay compensation	Ellen	200	...
3	35654521	30-12-2010:14.32	Register request	Pete	50	...
	35654522	30-12-2010:15.06	Examine casually	Mike	400	...
	35654524	30-12-2010:16.34	Check ticket	Ellen	100	...
	35654525	06-01-2011:09.18	Decide	Sara	200	...
	35654526	06-01-2011:12.18	Reinitiate request	Sara	200	...
	35654527	06-01-2011:13.06	Examine thoroughly	Sean	400	...
	35654530	08-01-2011:11.43	Check ticket	Pete	100	...
	35654531	09-01-2011:09.55	Decide	Sara	200	...
	35654533	15-01-2011:10.45	Pay compensation	Ellen	200	...
4	35654641	06-01-2011:15.02	Register request	Pete	50	...
	35654643	07-01-2011:12.06	Check ticket	Mike	100	...
	35654644	08-01-2011:14.43	Examine thoroughly	Sean	400	...
	35654645	09-01-2011:12.02	Decide	Sara	200	...
	35654647	12-01-2011:15.44	Reject request	Ellen	200	...
...	...	...	...	...	...	...

```
<?xml version="1.0" encoding="UTF-8" ?>
  <extension name="Concept" prefix="concept" uri="http://.../concept.xesext"/>
  <extension name="Time" prefix="time" uri="http://.../time.xesext"/>
  <extension name="Organizational" prefix="org" uri="http://.../org.xesext"/>
  <global scope="trace">
     <string key="concept:name" value="name"/>
  </global>
  <global scope="event">
     <date key="time:timestamp" value="2010-12-17T20:01:02.229+02:00"/>
     <string key="concept:name" value="name"/>
     <string key="org:resource" value="resource"/>
  </global>
  <classifier name="Activity" keys="concept:name"/>
  <classifier name="Resource" keys="org:resource"/>
  <classifier name="Both" keys="concept:name org:resource"/>
  <trace>
     <string key="concept:name" value="1"/>
     <event>
        <string key="concept:name" value="register request"/>
        <string key="org:resource" value="Pete"/>
        <date key="time:timestamp" value="2010-12-30T11:02:00.000+01:00"/>
        <string key="Event_ID" value="35654423"/>
        <string key="Costs" value="50"/>
     </event>
     <event>
        <string key="concept:name" value="examine thoroughly"/>
        <string key="org:resource" value="Sue"/>
        <date key="time:timestamp" value="2010-12-31T10:06:00.000+01:00"/>
        <string key="Event_ID" value="35654424"/>
        <string key="Costs" value="400"/>
     </event>
     <event>
        <string key="concept:name" value="check ticket"/>
        <string key="org:resource" value="Mike"/>
        <date key="time:timestamp" value="2011-01-05T15:12:00.000+01:00"/>
        <string key="Event_ID" value="35654425"/>
        <string key="Costs" value="100"/>
     </event>
     <event>
        <string key="concept:name" value="decide"/>
        <string key="org:resource" value="Sara"/>
        <date key="time:timestamp" value="2011-01-06T11:18:00.000+01:00"/>
        <string key="Event_ID" value="35654426"/>
        <string key="Costs" value="200"/>
     </event>
     <event>
        <string key="concept:name" value="reject request"/>
        <string key="org:resource" value="Pete"/>
        <date key="time:timestamp" value="2011-01-07T14:24:00.000+01:00"/>
        <string key="Event_ID" value="35654427"/>
        <string key="Costs" value="200"/>
     </event>

     <trace>
        <string key="concept:name" value="2"/>
        <event>
           <string key="concept:name" value="register request"/>
           <string key="org:resource" value="Mike"/>
           <date key="time:timestamp" value="2010-12-30T11:32:00.000+01:00"/>
           <string key="Event_ID" value="35654483"/>
           <string key="Costs" value="50"/>
        </event>
        ...
     </trace>
     ...
  </log>
```

Figura 65 Fragmento de un archivo XES [4]

XES soporta el concepto de clasificador. Un log XES define un número arbitrario de clasificadores. Cada clasificador se especifica por la lista de atributos. Si dos eventos tienen valores iguales para los respectivos atributos estos eventos son considerados iguales para ese clasificador. Estos atributos deberían ser atributos de eventos obligatorios. Hoy en día XES es soportado por herramientas como ProM, Nitro, XESame y OpenXES. XESame genera archivos XES desde una colección de tablas de bases de datos. La idea es

que dado un conjunto de tablas puede haber diferentes puntos de vista. Por lo tanto XES sirven como puntos de vistas en el log de eventos. OpenXES es la referencia de implementación de XES, una librería open source java para leer, almacenar y escribir logs XES. OpenXES se puede embeber fácilmente con otras herramientas y permite (des)serializar eficientemente log de eventos en/desde archivos XML. Esto libera a los desarrolladores de software de desarrollar código tedioso para importar y exportar datos de eventos.

Desafíos de extraer logs de eventos
Los cuatro desafíos son:

- Desafío 1: Correlación
  Los eventos dentro de un log de eventos deben estar agrupados por caso. Este requerimiento simple puede ser un desafío ya que requiere correlación de eventos, es decir, los eventos deben relacionarse unos con otros. ¿cómo identificar eventos con su caso correspondiente? Cuando se trabaja con sistemas de legado se requiere de un esfuerzo adicional para satisfacer la correlación.
- Desafío 2: Timestaps
  Los eventos se deben ordenar por caso. En principio cierto orden no requiere de timestamps. Sin embargo, cuando se mezcla datos de diferentes fuentes, se depende de los timestamps para ordenar eventos (en el orden de ocurrencia). Esto puede ser problemático por la variedad de los relojes y las demoras. Esto produce que el orden de los eventos en el log no sea confiable.
- Desafío 3: snapshots
  Los casos pueden tener un ciclo de vida que extienda el período en el que el log se generó. Por lo tanto es importante darse cuenta que el log de eventos por lo general provee un snapshots de un procesos que se encuentra corriendo. Cuando el promedio de duración de un caso es corto comparado con el largo del grabado, la mejor solución para este problema es remover los casos incompletos. Se remueven todos los casos que no tengan actividad inicial o final.
- Desafío 4: Alcance
  El cuarto problema es el alcance del log de eventos. Los sistemas de información de empresas pueden tener cientos de tablas con datos relevantes de negocio. ¿Cómo decidir que tablas incorporar? El conocimiento del dominio es fundamental para localizar los datos requeridos y el alcance de los mismos. Obviamente el alcance deseado depende de la disponibilidad de los datos y las preguntas que necesitan responderse.
- Desafío 5: Granularidad
  En muchas aplicaciones los eventos en el log de eventos tienen un distinto nivel de granularidad que las actividades relevantes para el usuario final. Algunos sistemas producen eventos de bajo nivel que son demasiado detallados para presentarse a un stakeholder interesado en administrar o mejorar procesos. Afortunadamente hay varias aproximaciones para preprocesar logs de eventos de bajo nivel.

A.2.1.2 Estructura básica de un documento XES

- Log
  En el nivel superior se encuentra el log, que contiene toda la información de eventos

relacionados a un proceso específico.
el tag xml es <log> como se ve en la Figura 66

Attribute key	Attribute type	Required	Description
xes.version	xs:decimal	Yes	The version of the XES standard the document conforms to (e.g., "1.0").
xes.features	xs:token	Yes	A whitespace-separated list of optional XES features this document makes use of (e.g., "nested-attributes"). If no optional features are used, this attribute must have an empty value.

Figura 66 el tag xml <log>

- Rastros
  El log contiene un numero arbitrario de rastros. Cada rastro describe la ejecución de una instancia específica, o caso, de los procesos del log.
  El tag xml es <trace>
- Evento
  Cada rastro contiene un numero arbitrario de eventos. Los eventos representan granos atómicos de una actividad que se ha observado durante la ejecución de un proceso.
  El tag xml es <event>
- Atributos
  el log, el rastro y los eventos no tienen información por ellos mismos. Solo definen la estructura del documento. Toda la información de log de eventos es almacenada en atributos. Los atributos describen a su elemento padre.
  Todos los atributos tienen una clave basada en string.
  Los, rastros y eventos contienen un numero arbitrario de atributos. Hay cuatro tipos de atributos, cada uno definido por el tipo de valor que representan: String, Date, Int, Float, Boolean.
- Atributos anidados
  Para permitir una flexibilidad máxima se permiten los atributos anidados. Ejemplo:

```
<string key=”city” value=”Bearlin”>
  <boolean key=”spellchecked” value=”false”/>
</string>
```

- El objeto log almacena dos listas de atributos globales para el nivel del rastro y para el nivel del evento. Los atributos globales son atributos que se entienden disponibles y propiamente definidos para cada elemento en su respectivo nivel a través del documento. Esto significa que el atributo global en el nivel de evento debe estar disponible para cada evento en el rastro. Un atributo global a nivel de rastreo debe estar disponible para cada rastro en el evento.

```
<global scope=”event”>
    <string key=”name” value=””/>
```

</global>
Los atributos globales son una característica requerida para el cumplimiento del estándar XES.

- Clasificador de eventos
  Los clasificadores de eventos son obligatorios en el estándar XES.
  Un clasificador de eventos se asigna a cada evento y lo identifica, lo que lo hace comparable con otros eventos (por medio del identificador asignado).
  Los clasificadores de definen por medio de un conjunto de atributos, de los cuales el identificador de una clase de un evento se deriva. En su forma más simple un clasificado se define con un solo atributo y el valor del atributo identificará la clase de un evento.
  Los clasificadores de eventos se definen por log, y puede haber un número arbitrario de clasificadores por documento. El conjunto de atributos usado para un clasificador debe ser un subconjunto de los atributos globales de ese log.
  Los clasificadores de eventos se definen con el tag xml <classifier>
  <classifier name="Activity classifier" keys="name status"/>
  El clasificador anterior define un clasificador con el nombre dado sobre los atributos de evento con clave: name y status. Cualquier par de eventos que tengan el mismo valor para estos dos atributos se consideran iguales para el clasificador.
- Extensiones
  Como no se define un conjunto específico de atributos por log, rastro, o evento, la semántica de los atributos que contienen estos elementos necesariamente es ambigua. Esta ambigüedad se resuelve por medio de las extensiones. Una extensión define un conjunto de atributos en cada nivel de la jerarquía del log XES (log, rastro, evento, y meta para atributos anidados). De esta forma se proveen puntos de interpretación de estos atributos. Las extensiones, por lo tanto, son principalmente un vehículo para adjuntar semántica a un conjunto de atributos definidos por elemento.
  Las extensiones tienen múltiples usos. Un uso importante es introducir un conjunto de atributos entendidos comúnmente que son vitales para una perspectiva específica o dimensión de un análisis del log de eventos. Hay un conjunto de extensiones estándar.
  El otro uso es incluye la definición de atributos generalmente comprendidos para un dominio de aplicación específico, o para soportar características específicas o requerimientos de una aplicación de análisis específico.
  <extension name="Concept" prefix="concept" uri="http://code.fluxicon.com/xes/concept.xesext"/>
  Uso:
  <string key="concept: name" value="Initialization"/>
  Las extensiones estándar son:
  Extension de concepto: para referirse al nombre del log o a la instancia de un evento en string.
  Extension del ciclo de vida: para los eventos específica la transición del ciclo de vida que representan en un modelo transaccional.
  Extensión organizacional: para expresar los recursos, el rol, o el grupo cuando los eventos pueden ser causados por actores humanos.
  Extension del tiempo: la fecha y hora exacta en que el evento ocurrió.

Para realizar *Process Mining*, los eventos se tienen que relacionar con casos. Un modelo de proceso describe el ciclo de vida de un tipo de caso particular. Todas las actividades en un modelo de proceso convencional (independientemente de la notación utilizada) corresponden a los cambios de estados de esos casos. Nos referiremos a esos modelos de procesos como modelos planos. Sin embargo es importante darse cuenta que los procesos en la vida real no son planos.
El log de eventos aplana la base de datos original que consiste de n tablas. El log plano es como una vista completa del conjunto de datos. Se pueden realizar vistas alternativas del conjunto original de datos.
Planificar un conjunto de datos en un log de eventos se puede comparar con agregar datos multidimensionales en Online Analytical Processing (OLAP), pero en *Process Mining* se analizan procesos más que un simple cubo OLAP. Por lo tanto, se necesita relacionar eventos y ordenarlos, lo cual hace el proceso de extracción más complejo.[3][4][6][9]

Anexo 3 Técnicas de *Process Mining*

En este anexo se profundizara sobre las distintas técnicas de *Process Mining*

A.3.1 Algoritmo α

El algoritmo α: dado un log de eventos simple produce una red de Petri que (por lo general) puede reproducir el log. Es el primer algoritmo de descubrimiento de procesos que puede representar la concurrencia. Sin embargo no es una técnica de minería muy práctica porque tiene problemas de ruido, comportamiento no frecuente o incompleto, y estructuras de ruteo complejas. El algoritmo α es simple y muchas de sus ideas fueron introducidas en técnicas más robustas y complejas como *Heurisitc Mining*, *Genetic Process Mining* y *Region Based Mining*.

La entrada de un algoritmo α es un log de eventos simple L, es decir $L \in B(A^*)$

Nos referiremos a conjuntos de actividades como A. Estas actividades corresponderán a transiciones en la red de Petri descubierta. Utilizaremos por convención las letras en mayúscula para referirnos al conjunto de actividades (e.g., $A,B \subseteq A$), mientras que para las actividades individuales minúscula (e.g., $a, b, c, \ldots \in A$). La salida del algoritmo α es una red de Petri. $\alpha(L) = (N,M)$. El objetivo es descubrir redes de workflow. Por lo tanto, podemos omitir el marcado inicial y escribir $\alpha(L) = N$ y el valor inicial de marcado es implícito $M = [i]$).

El algoritmo α escanea el log de eventos para descubrir patrones particulares. Por ejemplo, si la actividad a es seguida por la actividad b pero b nunca es seguida por a, entonces se asume que hay una dependencia casual entre a y b. Para reflejar esa dependencia, la red de Petri correspondiente debería mostrar la conexión entre a y b. Se distinguen cuatro relaciones de orden basadas en el log que tienen el objetivo de capturar los patrones relevantes del log.

Definición (relaciones de orden basadas en el log de eventos) Sea L un log de eventos sobre A , i.e.,

$L \in B(A^*)$. Sea $a, b \in A$:

• $a >_L b$ si y solo si hay un trace $\sigma = \_t_1, t_2, t_3, \ldots, t_n\_$ y $i \in \{1, \ldots, n-1\}$ entonces $\sigma \in L$ y $t_i = a$ y $t_{i+1} = b$

• $a \rightarrow_L b$ si y solo si $a >_L b$ y $b >_L a$

• $a \#_L b$ si y solo si $a >_L b$ y $b >_L a$

• $a \|_L b$ si y solo si $a >_L b$ y $b >_L a$

Considerar por ejemplo el log *L1 [(a,b,c,d)3, (a,c,b,d)2, (a,e,d)]*. Para este log de eventos se encontraron las siguientes relaciones de orden basadas en el log.

$>_{L1}$= {(a, b), (a, c), (a, e), (b, c), (c, b), (b, d), (c, d), (e, d)}

$\rightarrow_{L1}$= {(a, b), (a, c), (a, e), (b, d), (c, d), (e, d)}

$\#_{L1}$= {(a, a), (a, d), (b, b), (b, e), (c, c), (c, e), (d, a), (d, d), (e, b), (e, c), (e, e)}

$\|_{L1}$= {(b, c), (c, b)}

La relación $>L1$ contiene todos los pares de actividades en un relación de "seguimiento directo". $\rightarrow_{L1}$ contiene todos los pares de actividades en una relación de causalidad. Ejemplo, c→ d porque algunas veces d sigue directamente a c y nunca en el sentido contrario. $B\|L1\ c$ porque $b >_{L1} c$ y $c >_{L1} b$, es decir a veces c es seguida por b y otras veces en el sentido contrario. $b\#L1\ e$ porque $b />_{L1} e$ y $e/>_{L1} b$.

Para cualquier log de eventos se sostiene una de estas relaciones para cada par de actividades. Por lo tanto, la huella de un log se puede capturar en una matriz como la mostrada en la tabla 2

Tabla 13 Tabla de relaciones entre las actividades del log *L1 [4]*

	a	*b*	*c*	*d*	*e*
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

Las relaciones de orden basadas en el log se pueden usar para descubrir patrones en el modelo de proceso correspondiente como se muestra en la figura 5. Si *a* y *b* son una secuencia entonces el log tiene que mostrar la relación $a \rightarrow_L b$. Si luego de a hay una decisión entre *a* y *b*, el log mostrara $a \rightarrow_L b$, $a \rightarrow_L c$, y $b \#_L c$ porque *a* puede ser seguida por *b* y *c* pero *b* no va a ser seguida por *c* y viceversa. La contrapartida lógica de la compuerta XOR es el patrón de XOR-join se muestra en la figura 67 (b-c). Si $a \rightarrow_L c$, $b \rightarrow_L c$, y $a \#_L b$, esto sugiere que luego de la ocurrencia de *a* o *b, c* debe pasar. La figura 67 (d-e) muestra la compuerta AND y los patrones de AND-join. Si If $a \rightarrow_L b$,$a \rightarrow_L c$, y $b \|_L c$, entonces luego de *a*, se pueden ejecutar tanto *b* como *c* en paralelo. Si $a \rightarrow_L c$, $b \rightarrow_L c$, and $a \|_L b$, el log sugiere que *c* debe ser sincronizado con *a* y *b*.

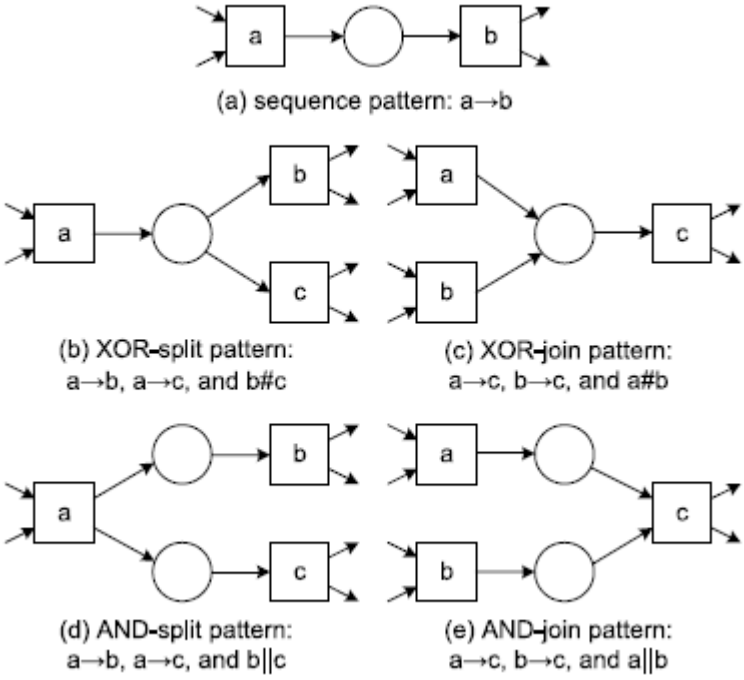


Figura 67 Patrones de procesos típicos y las huellas que dejan en un log de eventos [4]

Un ejemplo es con el log L3 en la figura 68

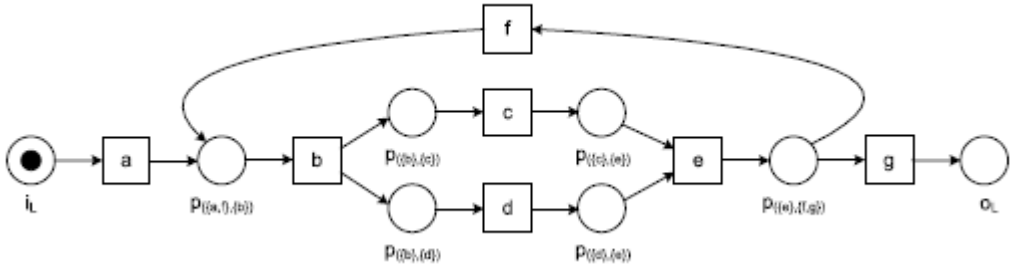


Figura 68 Red de workflow derivada del log L3 = [(a, b, c, d, e, f, b, d, c, e, g), (a, b, d, c, e, g)2, (a, b, c, d, e, f, b, c, d, e, f, b, d, c, e, g)] [4]

A.3.1.1Limitaciones del algoritmo α

Aunque asumiéramos que el log es completo, el algoritmo α tiene algunos problemas. Hay muchas redes de workflow distintas que pueden representar el mismo comportamiento, es decir, dos modelos pueden ser estructuralmente distintos pero equivalentes en el trace.

Por ejemplo para presentar el log:

$L_6 = [(a, c, e, g)_2, (a, e, c, g)_3, (b, d, f, g)_2, (b, f, d, g)_4]$

Se podría utilizar el modelo de la figura 69 aunque no es el más sencillo es capaz de representar los traces del log L6.

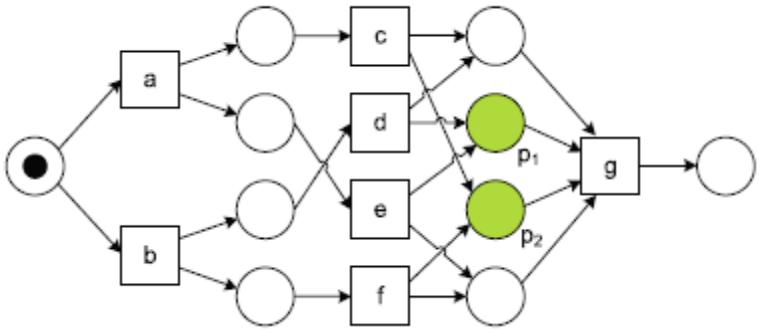


Figura 69 Red de workflow derivada de *L6*. Los dos estados resaltados son redundantes, es decir removiéndolos se simplificara el modelo sin cambiar su comportamiento. [4]

El algoritmo αoriginal tiene problemas con los loops cortos, es decir loops con longitud de uno o dos. Un ejemplo se puede ilustrar en la figura 70 que muestra el resultado de aplicar el algoritmo básico a L7.

$L_7 = [(a, c)_2, (a, b, c)_3, (a, b, b, c)_2, (a, b, b, b, b, c)_1]$

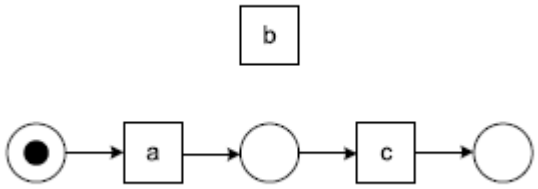


Figura 70 Red de workflow incorrecta [4]

El modelo resultante no es una red de workflow ya que la transición *b* esta desconectada del resto del modelo.

El modelo permite la ejecución de *b* antes de *b* y después de *c*. Esto no es consistente con el log de eventos. Utilizando una versión mejorada del algoritmo α este problema se puede solucionar y se puede descubrir un modelo como el de la figura 71.

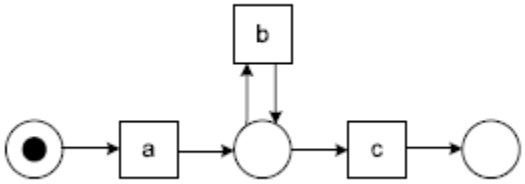


Figura 71 Red de workflow con un loop corto de largo uno [4]

Hay varias formas de mejorar el algoritmo α básico para que pueda tratar a loops. El algoritmo α mejorado usa una fase de pre y post procesamiento. En la fase de post procesamiento trata a los loops con largo 2 mientras que la fase de pre procesamiento inserta loops de largo uno.

El algoritmo básico no tiene problemas en descubrir loops de 2 o más actividades. Para un loop encontramos que $a >_L b$, $b >_L c$, y $c >_L a$. Si hay tres actividades concurrentes, encontramos a $>_L b$, $a >_L c$, $b >_L a$,

$b >_L c$, $c >_L a$, y $c >_L b$. Por lo tanto, es fácil detectar la diferencia. Para un loop de longitud dos, este no es el caso. Para un loop que comprende a y b, encontramos $a >_L b$ y $b >_L a$. Si a y b son concurrentes encontramos la misma relación. Por lo tanto, las dos construcciones dejan la misma huella en el log de eventos.

Un problema más difícil es el descubrimiento de las llamadas dependencias no locales que resultan de las construcciones del proceso de decisión no libre. En la figura 72 se ve un buen candidato luego de observar el siguiente log de eventos:

$$L_9 = [(a, c, d)_{45}, (b, c, e)_{42}]$$

Sin embargo el algoritmo α dará la red de workflow sin los lugares etiquetados con p1 y p2. Por lo tanto $\alpha(L_9) = N_4$, permitiendo los traces *(a,c,e)* y *(b,c,d)* aunque no aparezcan en L9. Estos problemas se pueden resolver utilizando otra versión del algoritmo α.

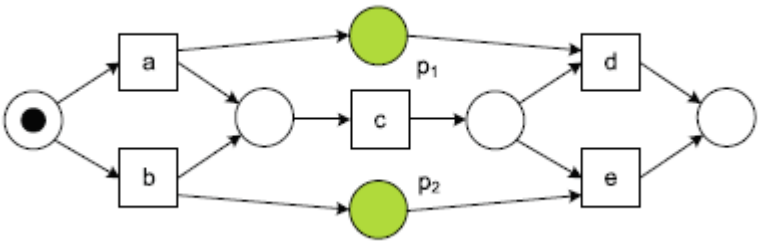


Figura 72 Red de workflow con una dependencia no local [4]

Otra limitación del algoritmo α es que no toma en cuenta las frecuencias. Por lo tanto el algoritmo es muy sensible al ruido y la incompletitud.

El algoritmo α es capaz de descubrir una gran clase de modelos. Varios de los problemas presentados en esta sección se pueden resolver haciendo refinamientos sobre el algoritmo. El algoritmo α garantiza la producción de un modelo de proceso correcto y puede ser descripto por una red de workflow que no contenga actividades duplicadas (dos transiciones con el mismo nombre de actividad) y transiciones silenciosas (actividades que no se registraron en el log) y no utilizan las dos construcciones mostradas en la figura 73

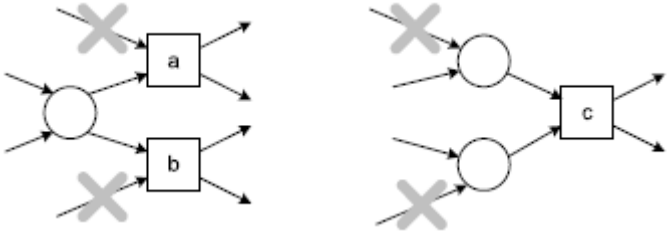


Figura 73 Dos construcciones que pueden poner en peligro la correctitud la red de workflow descubierta

Aunque el procesos a estudiar utilice construcciones como las mostradas en la figura 73 el algoritmo α producirá un modelo de proceso útil. [4]

La mayoría de los eventos tiene el atributo transaccional, por ejemplo $\#_{trans}(e)$ = *complete*. La extensión del ciclo de vida estándar de XES también provee ese atributo. El algoritmo α se puede adaptar fácilmente para tener en cuenta esta información. Primero de todo, el log se puede proyectar en log de eventos más pequeños en el que cada uno de los logs pequeños contenga todos los eventos relacionados a una actividad específica. Esta información se puede usar pada descubrir el ciclo de vida transaccional de cada actividad. Segundo, con respecto al proceso total, la información del ciclo de vida transaccional general o la información sobre una actividad específica se pueden explotar. En la figura 74 se ilustra la situación.

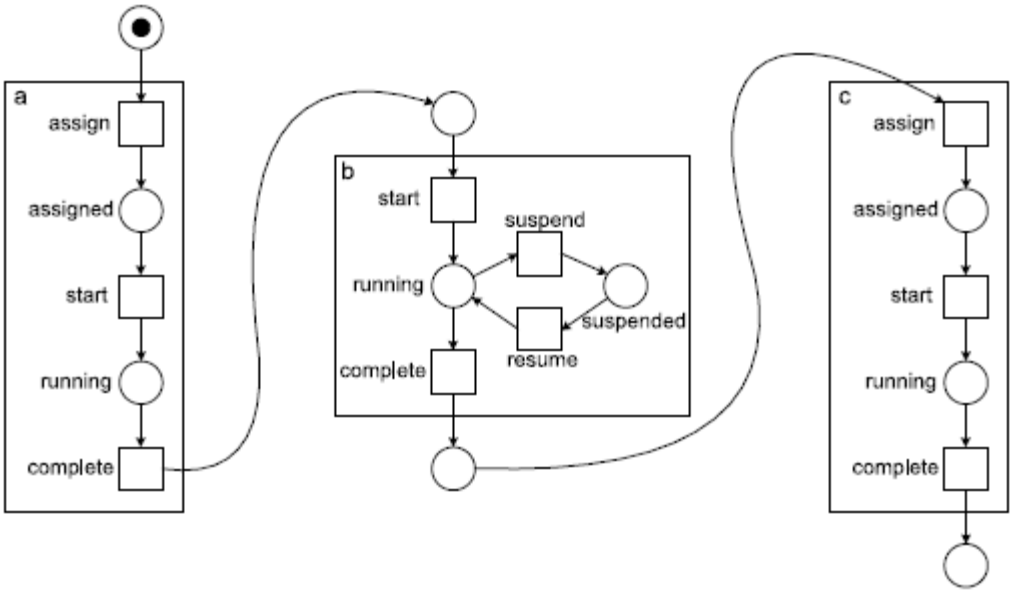


Figura 74 Hacer minería sobre log de eventos con información transaccional, el ciclo de vida de cada actividad se representa como un subproceso. [4]

Todos los eventos relacionados a una actividad esta mapeados en transiciones embebidas en subprocesos. Las relaciones entre las transiciones para cada subproceso son o descubiertas por separado o modeladas usando el conocimiento de dominio. En la figura 74 se muestran la secuencia de tres actividades. Las actividades *a* y *c* comparte el ciclo de vida transaccional común que involucran los tipos de eventos *assign, start, y complete*. La actividad *b* tiene un ciclo de vida transaccional que involucra los tipos de eventos *start, suspend, resume*, y *complete. [4]*

A.3.2 Heuristic Mining

Los algoritmos de heuristic Mining utilizan una representación similar a la de redes causales. Además estos algoritmos toman en cuenta frecuencias de eventos y secuencias cuando se construye un modelo. La idea básica es que los caminos no frecuentes no se deben incorporar al modelo. Tanto el sesgo representacional que proveen las redes causales como el uso de frecuencias hace al enfoque más robusto que otros. Un ejemplo de una red causal se ve en la figura 75

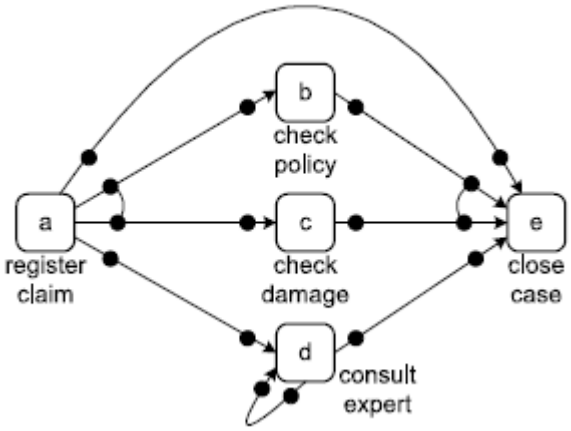


Figura 75 Red causal [4]

El proceso modelado por la figura 75 no se puede expresar por una red de workflow (WF-net, asumiendo que cada transición tiene un único nombre visible). Esto muestra que las C-nets (redes causales) son la representación más adecuada para el descubrimiento de procesos.

El objetivo de heuristic Mining es extraer una *C-net (causal net) C= (A, a_i, a_o,D, I,O)* del log de eventos. Los nodos del grafo de dependencia corresponden al conjunto de actividades *A*. Los arcos del grafo de dependencia corresponden a la relación de dependencia *D*. En una C-net solo hay una actividad de comienzo *ai*, y una actividad de fin *a0*. Se puede pre procesar el log e insertar comienzos y fines a cada trace. Por lo tanto el supuesto de que hay una única actividad de comienzo y una única actividad de fin no pone limitaciones parciales. De hecho es mejor tener un inicio y fin bien definido. También asumimos que en el grafo de dependencia todas las actividades están en el camino de *ai* a *a0*. No tiene sentido incluir actividades que no se encuentran en el camino de *ai* a *a0* esas actividades pueden estar muertas o pueden estar activas antes de que inicie el caso y no contribuyen a la completitud del caso. Por lo tanto asumimos que por medio de la construcción del grafo de dependencia, ya tenemos la estructura de un C-net. Por lo tanto solo las funciones $I \in A \rightarrow AS$ y $O \in A \rightarrow AS$ se tienen que derivar para completar la C-net.

Por medio de la reproducción de log de eventos en el grafo de dependencia, podemos estimar los enlaces de entrada y salida. Utilizando umbrales, es posible excluir enlaces basados en sus frecuencias. Esto resulta en funciones *I* y *O,* que completan la C-net.

Como se muestra *O(a)* = {{*b, c*}, {*d*}, {*e*}} y *I (e)* = {{*a*}, {*b, c*}, {*d*}}. Los enlaces {*b*} y {*c*} no están incluidos en *O(a)* y *I (e)* porque ocurren solo una vez (por debajo del umbral). La figura 76 muestra también la frecuencia de las actividades, dependencias, y enlaces. Por ejemplo la actividad ocurre 40 veces. El enlace de salida {*b, c*} de *a* ocurre 20 veces, etc.

Cada nodo muestra la frecuencia de la actividad correspondiente. Cada arco tiene una frecuencia que muestra cada cuanto ambas actividades se linkean. Las frecuencias links de entrada y salida también se representan, por ejemplo 20 de las 40 ocurrencias de *a* fueron seguidas por la ejecución de *b* y *c*.

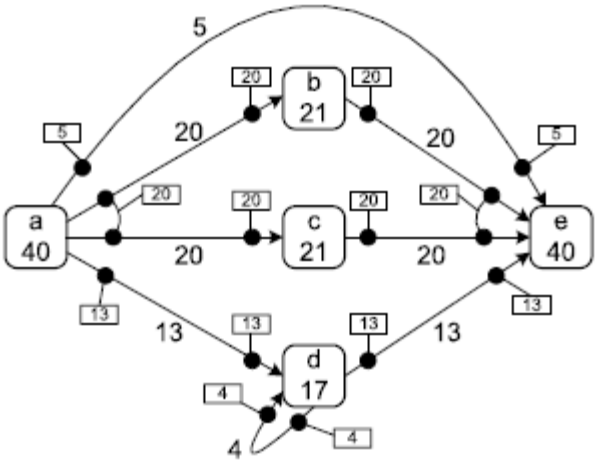


Figura 76 C-net derivada del log de eventos L. ***[4]***

En la figura 77 hay una visualización más intuitiva del C-net de la figura 676- Ahora el grosor de los arcos corresponde a las frecuencias de los caminos correspondientes. Esta visualización es importante para obtener una visión de los principales flujos del proceso.

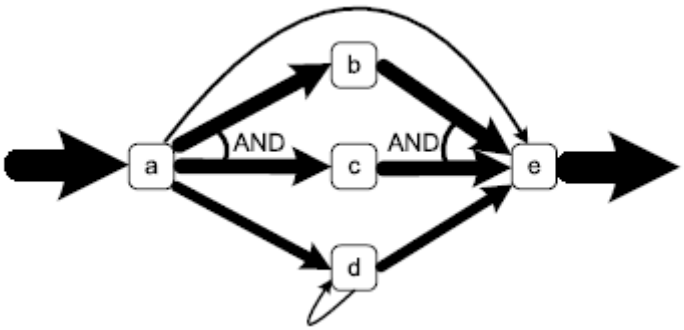


Figura 77 Visualización alternativa de la C-net mostrando claramente los caminos más tomados en el modelo de proceso. [4]

Varios de los fundamentos explicados en esta sección pueden ser aplicados a fuzzy Mining. Este enfoque provee de un conjunto extensible de parámetros para determinar qué actividades y arcos deben ser incluidos. Además, el enfoque puede construir modelos jerárquicos es decir, las actividades menos frecuentes se pueden mover a subprocesos. Además la metáfora del mapa de camino se explota para crear modelos de procesos que se puedan entender fácilmente mientras se provea información de la frecuencia y la importancia de las actividades y los caminos. [4]

A.3.3 *Process Mining* genética

El algoritmo α y las técnicas de Heuristic y Fuzzy Mining proveen modelos de procesos de una manera directa y determinística. Enfoques evolucionarios utilizan un procedimiento iterativo para mimetizar la evolución natural del proceso. Esos enfoques no son determinísticos y depende de aleatoriedad para encontrar nuevas alternativas. Veremos como ejemplo un enfoque de descubrimiento de procesos que utiliza una técnica del campo de la inteligencia computacional.

En la figura 78 se muestra una vista general del enfoque usado. Como en un algoritmo de genética hay cuatro caminos principales (a) inicialización, (b) selección, (c) reproducción y (d) terminación.

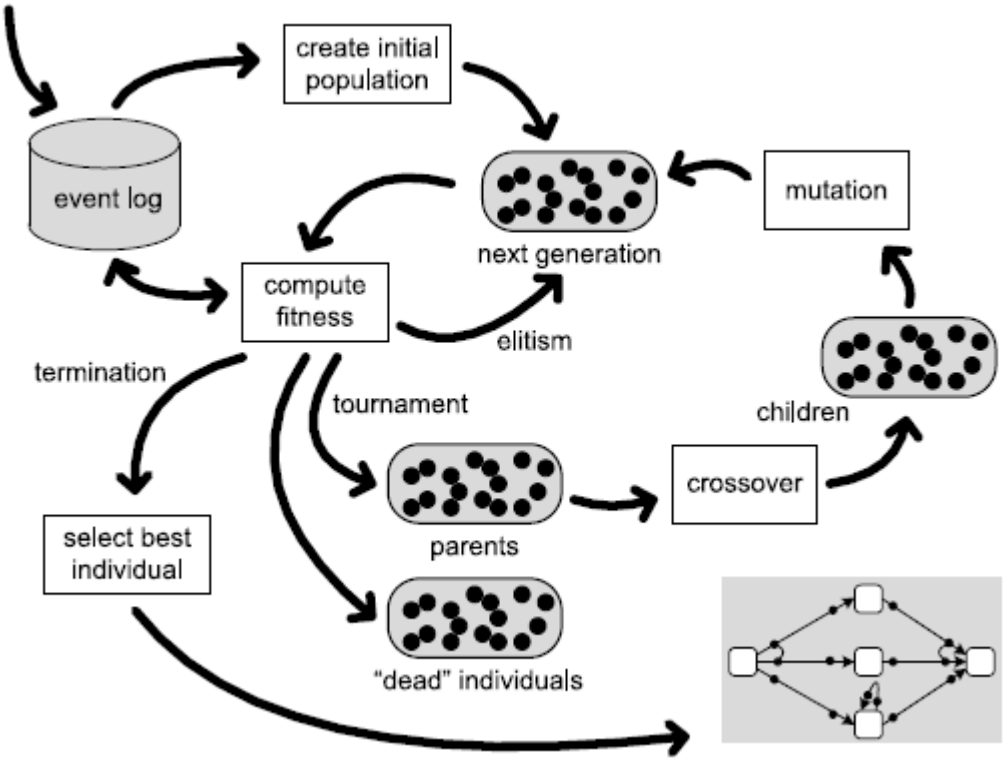


Figura 78 Visión general del enfoque que usa *Process Mining* genética. [4]

En el paso de inicialización, la población inicial se crea. Esta es la primera generación de individuos que se usaran. Cada individuo es un modelo de proceso. Usando los nombres de actividades que aparecen en el log, los modelos de procesos se crean aleatoriamente. Puede haber cientos o incluso miles de individuos en cada generación. Los modelos de procesos (es decir individuos) en la población inicial pueden tener poco que ver con el log de eventos, las actividades son las mismas pero el comportamiento del modelo inicial suele ser muy diferente al comportamiento que se ven en el log. Sin embargo, por accidente los modelos generados pueden tener partes que se adecuen al log de eventos por efectos aleatorios y el alto número de individuos.

En el paso de selección, se computariza e fitness de cada individuo. Una función de fitness determina la calidad del individuo en relación al log. Una forma de medir la calidad del modelo es la proporción de traces en el log que se pueden reproducir en el modelo. Esta no es una buena función de fitness, porque es muy probable que ninguno de los modelos en

la población inicial pueda reproducir los traces del log de eventos. Además, usando este criterio el modelo general como "el modelo flor" tendría alto fitness. Por lo tanto se necesita una función de fitness más refinada, una que tenga en cuenta la correctitud parcial del modelo y que tome en cuenta los cuatro criterios de calidad. Los mejores individuos, es decir los modelos de proceso que tienen el fitness mayor se mueven a la próxima generación. Esto se llama elitismo. Por ejemplo, el 1% de los mejores de la generación actual pasan a la próxima generación sin modificaciones. A través de torneos, se seleccionan padre para crear nuevos individuos. Los torneos entre individuos y elitismo deberían asegurarse de que "el material genético de los mejores modelos de procesos" tiene la mayor probabilidad de usarse para la próxima generación: supervivencia del que tenga más fitness. Los individuos con fitness pobre no sobreviven. En la figura 78 son individuos muertos.

En la fase de reproducción, los individuos emparentados seleccionados se usan para crear nuevas descendencias. Aquí se utilizan dos operadores genéticos: cruce y mutación. Para el cruce dos individuos se toman y se usan para crear dos nuevos modelos, esto termina en un pool con modelos hijos mostrados en la figura 78. Estos modelos hijos comparten partes del material genético de sus padres. Los hijos resultantes son los dos modificados utilizando mutación, por ejemplo aleatoriamente agregando o eliminando una dependencia causal. La mutación se utiliza para agregar nuevo material genético a la próxima generación.

A través de la reproducción (cruce y mutación) y elitismo, se crea una nueva generación. Para los modelos en esta generación se computa el fitness. De nuevo los mejores individuos se mueven a la próxima ronda (elitismo) o son usados para producir una nueva descendencia. Esto se repite y se espera que la calidad de cada generación sea mejor. El proceso de evolución termina cuando se encuentra una solución satisfactoria, es decir un modelo que tenga al menos el fitness deseados. Dependiendo del log de eventos puede tomar mucho tiempo para que el algoritmo converja. De hecho, gracias al sesgo representacional y el ruido en el log de eventos puede que no haya un modelo que tenga el nivel deseado de fitness. Por lo tanto, otro criterio de terminación se puede agregar. (Parar cuando se hayan hecho 10 generaciones sin tener mejoras) Cuando se finaliza se obtiene un modelo con el mejor fitness.

El enfoque descripto en la figura 78 es muy general. Cuando se implementa *Process Mining* genética se deben tomar las siguientes decisiones de diseño:

- *Representación de los individuos*: es importante determinar la clase de procesos que se descubrirán (sesgo representacional).
- *Inicialización:* para la población inicial los modelos se deben generar aleatoriamente. Dos enfoques propuestos (a) un enfoque en el que con una probabilidad determinada una dependencia causal entre dos actividades se inserta para crear la C_net. Y (b) un enfoque en el que una variante aleatoria de heuristic Mining se usa para crear la población inicial con el fitness mas alto promedio que las C-nets generadas aleatoriamente.
- *Función de fitness:* definir una función que balancee los cuatro criterios de calidad. La función de fitness conduce la evolución de los procesos y se puede usar para favorecer modelos particulares.

- *Selección de estrategia* (torneo y elitismo): el algoritmo genético tiene que determinar la fracción de individuos que deben ir a la próxima vuelta sin cambios.
- *Curse:* el objetivo es combinar material genético existente. La idea básica es crear un nuevo modelo de proceso que use partes de sus dos modelos padres.
- *Mutación:* aleatoriamente insertar material genético nuevo.

En la figura 79 se muestra un ejemplo de cruce y en la 40 un ejemplo de mutación.

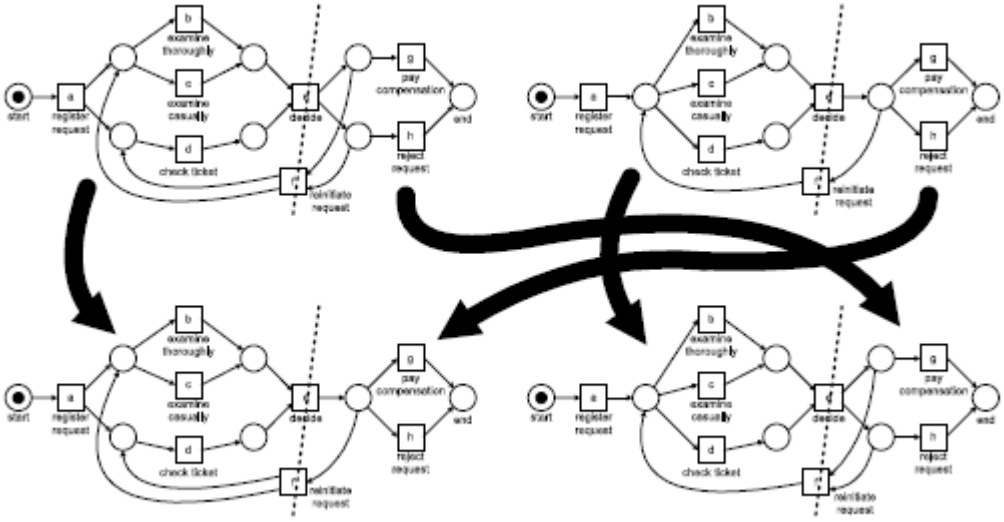


Figura 79 Dos modelos padre (arriba) y dos modelos hijo resultado del cruce. Los puntos de cruce están dados por las líneas punteadas [4]

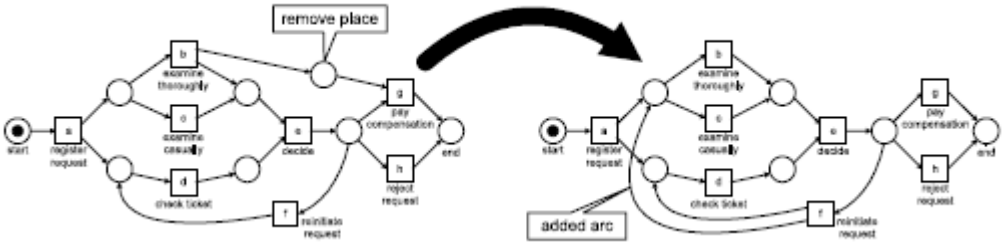


Figura 80 Mutación: un sitio es removido y un arco se agrega [4]

Genetic *Process Mining* es flexible y robusta. Como las técnicas de heuristic Mining puede tratar con ruido y logs que no estén completos. El enfoque se puede adaptar y extender fácilmente. Como en la mayoría de los enfoques evolutivos, genetic *Process Mining* no es muy eficiente para modelos grandes. Puede tomar mucho tiempo descubrir un modelo que tenga un fitness aceptable. En teoría se puede mostrar que eligiendo operadores genéticos adecuados se garantiza que eventualmente se producirá un modelo con un fitness óptimo. Una ventaja que tiene es que es fácil proveer implementaciones paralelas de *Process Mining* genética. Es útil combinar *Process Mining* genética y heuristic Mining . En este caso *Process Mining* genética se usa para mejorar el modelo de proceso obtenido por heuristic Mining. Esto puede resultar en modelos que no podrían haber sido descubiertos

nunca con el algoritmo convencional buscando solo dependencias locales y en un ahorro de tiempo computacional. [4]

A.3.4 Minería basada en regiones

Las regiones basadas en estado se pueden usar para construir una red de Petri desde un sistema de transición. Las regiones basadas en lenguajes se pueden usar para construir una red de Petri desde un lenguaje con prefijo cerrado. Se pueden aplicar enfoques sintéticos usando regiones basadas en el lenguaje directamente al log de eventos. Para aplicar regiones basadas en estado primero se tiene que crear un sistema de transiciones.

A.3.5 Sistemas de transiciones

Para construir una red de Petri usando regiones basadas en estado, primero necesitamos descubrir un sistema de transiciones basado en los traces del log de eventos. Los sistemas de transiciones se pueden describir como $TS = (S,A,T)$ donde S es el conjunto de estados, $A \subseteq \mathcal{A}$ es el conjunto de actividades y $T \subseteq S \times A \times S$ es el conjunto de transiciones. $S_{start} \subseteq S$ es el conjunto de estados iniciales. $S_{end} \subseteq S$ es el conjunto de estados finales.

Definición: Sistema de transiciones basada en un log de eventos. Sea $L \in \mathcal{B}(\mathcal{A}^*)$ un log de eventos y $l_{state}()$ una función de representación de estado. $TS_{L,l_{state}()} = (S,A,T)$ es un sistema de transición basado en L y $l_{state}()$ con:

- $S = \{l_{state}(\sigma, k) \mid \sigma \in L \wedge 0 \leq k \leq |\sigma|\}$ es el espacio de estado.
- $A = \{\sigma(k) \mid \sigma \in L \wedge 1 \leq k \leq |\sigma|\}$ es el conjunto de actividades.
- $T = \{(l_{state}(\sigma, k), \sigma(k+1), l_{state}(\sigma, k+1)) \mid \sigma \in L \wedge 0 \leq k < |\sigma|\}$ es el conjunto de transiciones.
- $S_{start} = \{l_{state}(\sigma, 0) \mid \sigma \in L\}$ es el conjunto de estados iniciales.
- $S_{end} = \{l_{state}(\sigma, |\sigma|) \mid \sigma \in L\}$ es el conjunto de estados finales.

Consideremos el log $L1 = [(a, b, c, d)_3, (a, c, b, d)_2, (a, e, d)]$. La figura 81 muestra el sistema de transición $TS_{L1,l_{state1}()}$. Si consideramos por ejemplo un caso con un trace $\sigma = (a, b, c, d)$. Al principio el caso está en un estado $l_{state1}(\sigma, 0) = ()$. Luego de ejecutar a el caso está en el estado $l_{state1}(\sigma, 1) = (a)$. Luego de ejecutar b el estado es $l_{state1}(\sigma, 2) = (a, b)$ y así sucesivamente. Los cinco estados visitados por el caso se agregan al sistema de transición. Las correspondientes transiciones se agregan también. Lo mismo se realiza con los otros casos y se obtiene como resultado el sistema de transiciones de la figura 81.

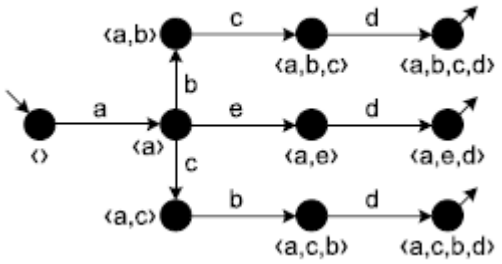


Figura 81 Sistema de transiciones del log *L1 [4]*

Hasta ahora, solo consideramos un log de eventos simple como entrada. Los eventos de la vida real contienen mucha más información. Información de recursos y de datos también se puede tener en cuenta cuando se construye un sistema de transiciones. Esta información se puede usar para identificar estados y etiquetar transiciones.

Un sistema de transiciones define un modelo de proceso de bajo nivel. Desgraciadamente, esos modelos no pueden expresar construcciones de alto nivel y sufren del problema de explosión de estado. De un proceso simple con 10 actividades paralelas se obtendrán 1024 estados y 5120 transiciones. Afortunadamente las regiones basadas en estado se puedan usar para sintetizar un modelo más compacto de un sistema de transiciones.

A.3.6 Descubrimiento de procesos usando regiones basadas en estado

Luego de transformar un log de eventos en un sistema de transición de bajo nivel, podemos sintetizar una red de Petri de él. Esta red de Petri se puede utilizar para construir modelos de procesos en otra notación de alto nivel (ejemplo BPMN, UML, etc.). El objetivo es plegar un sistema de transiciones grande en una red de Petri más pequeña detectando concurrencia. La idea principal es descubrir regiones que correspondan a lugares. Una región es un conjunto de estados de manera que todas las actividades en el sistema de transición "concuerdan" en la región.

Definición (Regiones basadas en estado) Sea $TS = (S,A,T)$ un sistema de transición y $R \subseteq S$ un subconjunto de estados. R es una *región* si para cada actividad $a \in A$ se mantiene una de las siguientes condiciones:

1. Todas las transiciones $(s_1, a, s_2) \in T$ *entran* R, es decir, $s_1 \notin R$ y $s_2 \in R$.
2. Todas las transiciones $(s_1, a, s_2) \in T$ *salen* R, es decir, $s_1 \in R$ y $s_2 \notin R$.
3. Todas las transiciones $(s_1, a, s_2) \in T$ *no cruzan* R, es decir, $s_1, s_2 \in R$ o $s_1, s_2 \notin R$.

Sea R una región. En el caso de que todas las actividades se puedan clasificar en entrar, salir y no cruzar la región. Una actividad no puede entrar en la región en una parte del sistema de transición y luego salir de la región en otra parte. En la figura 82 se muestra este concepto. El rectángulo punteado describe una región R, es decir, un conjunto de estados en el sistema de transiciones. Todas las actividades tienen que tomar una posición con respecto a esta región. Todas las transiciones nombradas a entran en la región

R. Si habría una transición con nombre *a* que no conecta con un estado de afuera de la región con un estado dentro de la región entonces *R* no sería una región. Todas las transiciones nombradas *b* entran a la región todas las llamadas *c* y d salen de la región. Todas las transiciones nombradas *e* y *f* no cruzan con R, es decir, siempre conectan dos estados de afuera de la región o dos estados dentro de la región.

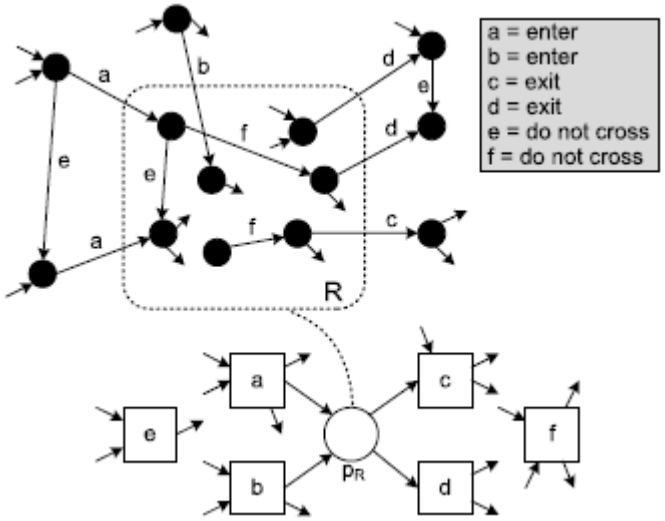


Figura 82 La región R corresponde al sitio *Pr*. Todas las actividades se pueden clasificar en entrada a la región (*a* y *b*), salida de la región (*c* y *d*) y en no cruzan la región (*e* y *f*) [4]

Por definición la unión de dos regiones es de nuevo una región. Por lo tanto solo nos interesan las regiones mínimas. La idea básica es que cada región mínima *R* corresponde a un lugar *pr* en la red de Petri como se muestra en la figura 82. Las actividades que entran en la región se convierten en transiciones en la red de Petri que tienen *pr* como el lugar de llegada, las actividades que dejan la región se transforman en transiciones de salida de *pr* y las actividades que no cruzan la región corresponden a transiciones en la red de Petri que no se conectan con *pr*. Por lo tanto solo pocas regiones corresponden a la res de Petri.

Un ejemplo claro se ilustra en la figura 83. Esta figura muestra un pequeño proceso con poca concurrencia por lo tanto el sistema de transiciones y la red de Petri tienen tamaños similares. Sin embargo para procesos grandes con mucha concurrencia la reducción puede ser considerable. Un sistema de transición modelado 10 actividades paralelas con $2_{10} = 1024$ estados y $10 \times 2_{10-1} = 5120$ transiciones se puede reducir a una red de Petri con solo 20 lugares y 10 transiciones. [4]

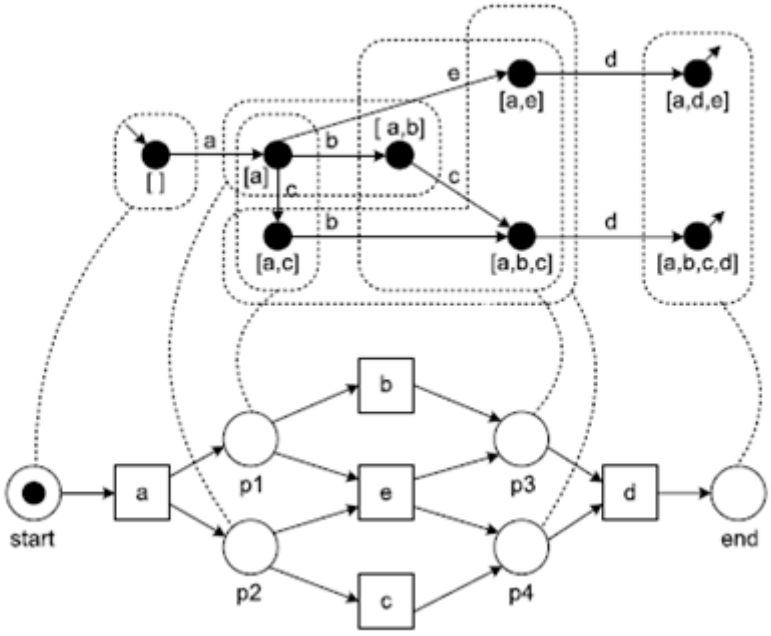


Figura 83 Sistema de transiciones derivado del log *L1*= [(*a, b, c, d*)3, (*a, c, b, d*)2, (*a, e, d*)], se convierte en una red de Petri usando regiones basadas en estado. [4]

A.3.7 Chequeo de Concordancia: Reproducción de tokens para medir fitness

Las medidas fitness: "la proporción del comportamiento en el log de eventos que concuerda posiblemente con el modelo". De los cuatro criterios, el criterio de fitness es el más relacionado con l concordancia.

El fitness de un caso con trace σ en una red de workflow N se define como:

$$fitness(\sigma, N) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

Una forma de computar el fitness es a través de la computación *naïve fitness*, en la que se deja de reproducir un trace una vez que se encuentra un problema y se marca esta como notificación. Consideremos ahora otra técnica en la que continuaremos reproduciendo los traces en el modelo pero registrando todas las situaciones en las que una transición está obligada a dispararse sin ser habilitada, es decir se cuentan la cantidad de tokens perdidos.

La primer parte de la ecuación calcula la fracción de los tokens perdidos (m : *missing*)con respecto al número de tokens consumidos (c : *consumed*). Si no hay tokens perdidos *m=0* y si *1- m/c =0* entonces todos los tokens consumidos se perdieron (m=c). De la misma forma *1-r/p = 1* si no hay tokens restantes y *1-r/p = 0* si ninguno de los tokens producidos fue realmente consumido.

Al reproducir el log de eventos se cheque los tokens que se perdieron, por distintos motivos ya sea por q debía ejecutarse una actividad y no se ejecutó, se deben realizar anotaciones en el modelo indicando las discrepancias con la realidad y cuales tareas para cuantos casos no fueron ejecutadas.

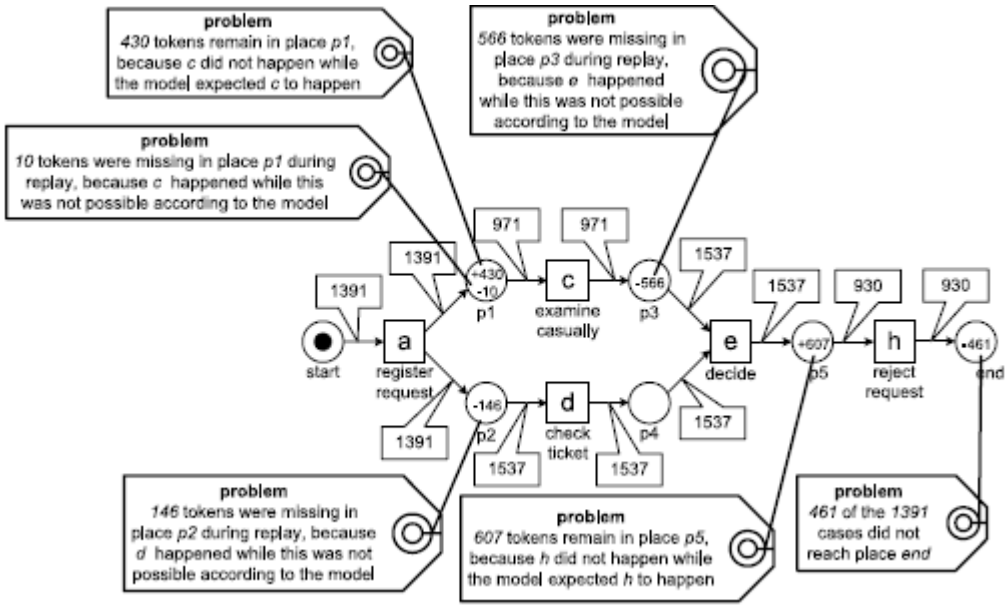


Figura 84 Información de diagnóstico que muestra las desviaciones (*fitness (Lfull,N3)*= 0.8797) [4]

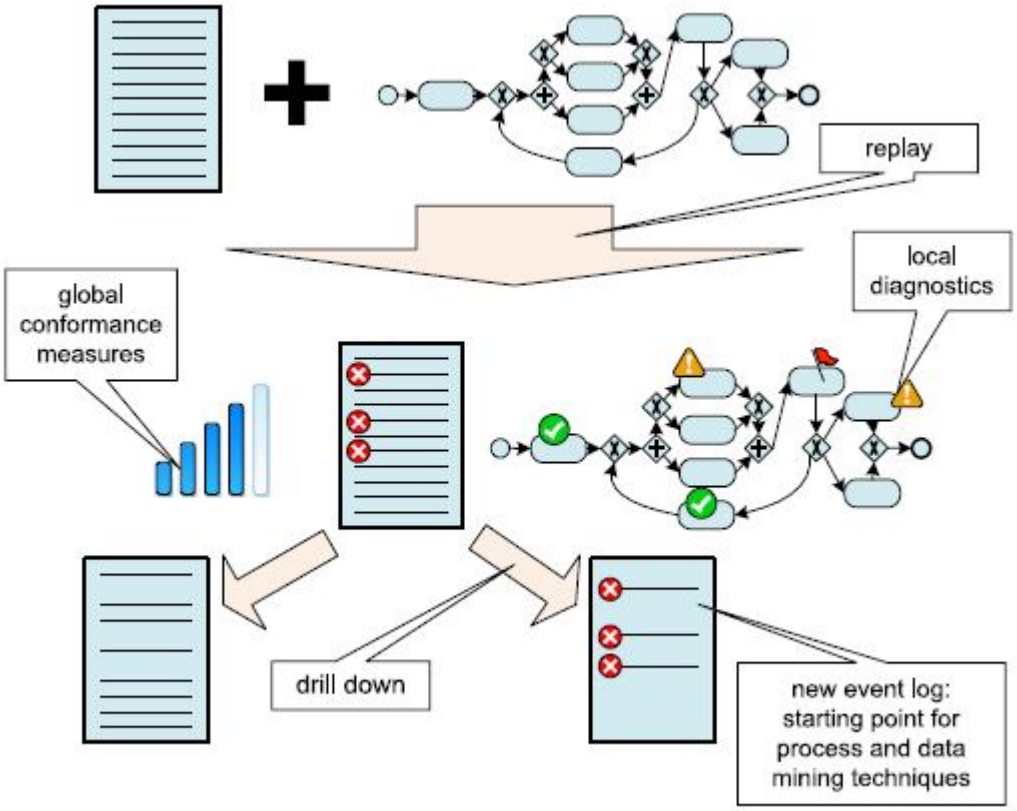


Figura 85 El chequeo de concordancia provee medidas globales de concordancia como fitness y diagnósticos locales. Además el log de eventos se particiona en los casos que se adecuan al modelo y los que no. Los dos sublogs se pueden utilizar para un análisis futuro. [4]

Como se muestra en la figura 85 un log de eventos se puede separar en dos sublogs: un log de eventos conteniendo sólo los casos que se adecuan y otro log de eventos que

contiene los casos que no se adecuan al modelo. Cada uno de los logs se puede utilizar para análisis futuros. Por ejemplo uno puede construir un modelo de procesos para el log de eventos que contiene los casos desviados. También se pueden utilizar otras técnicas de data y de *Process Mining*. Por ejemplo, es interesante conocer que personas estaban involucradas en los casos desviados y si los casos tardaron más o fueron más costosos. También se pueden utilizar técnicas de clasificación para poder investigar cuando no haya concordancia.
La idea de reproducir un log de eventos sobre un modelo de proceso no se limita a redes de Petri. Cualquier notación para modelado de proceso con semántica ejecutable permite la reproducción del log de eventos. Sin embargo, tener lugares explícitos y un inicio y fin claros en una WF-nets facilita la generación de diagnóstico con significado. La reproducción se vuelve más complicada cuando hay actividades duplicadas y silenciosas. Por lo general puede haber relaciones muchos a muchos entre los nombres de los eventos en el log de eventos y nombres de actividades en el modelo de proceso. Las actividades que aparecen en el log pero que no tienen su contraparte en el modelo son fáciles de manejar, uno simplemente puede descartar estos eventos. Para manejar actividades duplicadas o silenciosas, se utilizan exploraciones de estado local para encontrar el camino correspondiente (más adecuado) en el modelo de proceso. Por ejemplo las transiciones etiquetadas -t en el modelo que no corresponden a eventos en el log se ejecutan solamente si pueden habilitar transiciones que corresponden a eventos subsecuentes en el log de eventos. El inconveniente es que esas exploraciones de estado local pueden producir consumición de tiempo para log de eventos.
Las técnicas vistas pueden tratar con la situación de que un conjunto de actividades del modelo Am difiera del conjunto de actividades del log Al. Las actividades en el log pero no en el modelo (Al \ Am) son simplemente ignoradas y las actividades en el modelo pero no en el log (Am\ al) se silencian. Esto puede dejar valores de concordancia más optimistas. Como alternativa uno también puede pre procesar el modelo y el log para que de esta forma los dos coincidan en el conjunto de actividades. Después del paso de pre procesamiento, los eventos que se refieren a actividades que no aparecen en el modelo y el salteo de actividades que no aparezcan en el log se consideran desviaciones.
Anteriormente nos centramos exclusivamente en fitness (es decir la porción de eventos en el log de eventos que puede ser explicado por el modelo de proceso) Esta es solo una de las cuatro criterios de calidad. Para el chequeo de concordancia el resto de los criterios de calidad son menos relevantes. Sin embargo, las técnicas de replay se pueden utilizar también para análisis de precisión (esquivando modelos con bajo fittnes) y generalización (esquivando modelos con mucho fitness). Si en promedio, muchas transacciones se habilitaron durante la reproducción, el modelo se dice ser de bajo fittnes. Si en promedio, muy pocas transiciones se habilitaron durante la reproducción, el modelo se dice ser sobre fittnes. [4]

A.3.7.1 Otras técnicas de chequeo de concordancia

El chequeo de concordancia se puede usar para mejorar el alineamiento de procesos de negocio, organizaciones y sistemas de información. Como se mostró las técnicas de Replay y análisis footprint ayudan a identificar diferencias entre el modelo de proceso y el proceso real almacenado en el log de eventos. Las diferencias identificadas pueden llevar a cambios en el proceso o en el modelo. Por ejemplo, exponer desviaciones entre el modelo y

el proceso puede llevar a mejores instrucciones de trabajo o cambios en la administración. El chequeo de concordancia es también una poderosa herramienta para auditores que necesitan asegurarse que los procesos se ejecutan dentro de los límites seteados por varios stakeholders.
El chequeo de concordancia se puede utilizar para otros propósitos como reparar modelos y la evaluación de algoritmos de descubrimiento.

Reparación de modelos

Cuando un modelo de proceso y un log de eventos no concuerdan en el proceso, esto lleva a adaptaciones del modelo o del proceso en sí mismo. Asumamos que queremos usar chequeo de concordancia para reparar el modelo, es decir para alinearlo con la realidad. Las técnicas de Token Replay se pueden usar para reparar los modelos de forma semi automática. Por ejemplo los caminos que nunca son tomas se pueden remover. Token replay no ayuda a remover concurrencia que no se utiliza nunca (es decir actividades que se modelan en paralelo pero se ejecutan secuencialmente). Sin embargo, esto se puede observar con la matriz de footprint. Luego de remover las partes q no se utilizan del modelo los tags *m* y *r* que apuntan a tokens perdidos o restantes se pueden utilizar para reparar el modelo. Un tag *m* indica actividades que sucedieron en el proceso pero que no debían suceder de acuerdo al modelo. Un tag indica actividades que no sucedieron pero que deberían haber sucedido de acuerdo al modelo. Comparando las matrices de footprint del log y del modelo se mostrarán problemas similares. Esta información la puede utilizar un diseñador para reparar el modelo. En principio, esto se puede realizar automáticamente.

Evaluación de algoritmos de descubrimiento de procesos

El chequeo de concordancia se puede usar también para evaluar y comparar técnicas de descubrimiento de procesos. Aunque el foco de la técnica de reproducción de token es la dimensión de finttnes, se puede utilizar la técnica para las dimensiones de precisión y generalización. Un modelo con una falta severa de precisión, tendrá en promedio muchas transiciones habilitadas durante la reproducción. Si, en promedio, muy pocas transiciones fueron habilitadas durante la reproducción, el modelo por lo general tiende a tener overfittnes y por lo tanto no es suficientemente general. También se puede usar la matriz de footprint para analizar precisión y generalización.

Conectando el log de eventos y el modelo de proceso

Cuando se reproduce un modelo, los eventos en el log se relacionan con las actividades en el modelo, es decir el log de eventos se conecta con el modelo de proceso. Relacionando actividades con eventos, la información extraída del log de eventos se puede usar para enriquecer al proceso. Por ejemplo timestamps en el log de eventos se puede usar para hacer declaraciones de la duración de una actividad modelada.

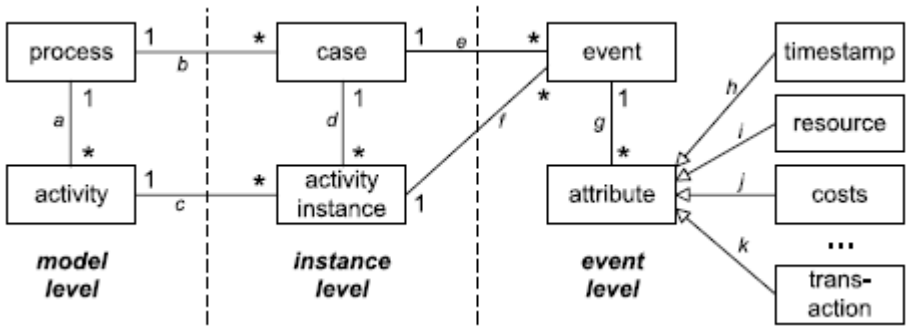


Figura 86 El nivel de instancia- establecido durante el chequeo de concordancia- conecta el nivel de modelo y el nivel de evento [4]

La figura 86 muestra la conexión entre el modelo de proceso y el log de eventos que se realiza durante la reproducción del modelo. Se identifican tres niveles: nivel de modelo, nivel de instancia y nivel de eventos. Los modelos de proceso y de eventos por lo general existen independientemente uno del otro, una excepción son sistemas de WFM y BPM en los que la conexión entre estos modelos existe de antemano. Es por esto que estos sistemas no son tan importantes para la automatización de procesos, pero si sirve como poderoso habilitador de soluciones de *Business Intelligence*. Sin embargo, en la mayoría de los proceso, no hay soporte de sistemas WFM o BPM. Como resultado, los modelos de proceso (si existen) y los log de eventos están muy vagamente acoplados. Afortunadamente la técnica de token replay se puede usar para establecer una fuerte relación entre los niveles de modelo y eventos.
Durante la reproducción los eventos se conectan con instancias de actividad. De esta forma el log de eventos les da vida de otra forma a modelos de proceso estáticos. Como resultado toda la información extraída del log de eventos se puede proyectar en el modelo por ejemplo mostrando cuellos de botella y sobresaltando caminos frecuentes.
Los atributos proveen de información valiosa cuando se conectan con las actividades. Por ejemplo, timestamps se puede usar para visualizar cuellos de botella, tiempos de espera etc. Los datos de recursos adjuntados a los eventos se pueden usar para aprender patrones de trabajo y reglas de alocación. La información de costos se puede ver proyectada en los modelos de proceso para ver ineficiencias.

A.3.8 Extension del modelo (otras perspectivas)

El primer paso en cualquier proyecto de *Process Mining* es obtener conocimiento del proceso y de los datos en el log de eventos. El llamado *dotted chart* provee una vista de helicóptero del proceso. En un *dotted chart*, cada evento se describe como un punto en un plano de dos dimensiones como se muestra en la figura 87 El eje horizontal representa el tiempo del el evento. El eje vertical representa la clase del evento. Para determinar la clase de un evento usamos el clasificador. Un clasificador es una función que mapea los atributos de u evento en una etiqueta, es la clase del evento. Un ejemplo de un clasificador es $e = \#_{case}(e)$, es decir el id del caso del evento. Otro ejemplo pueden ser $e = \#_{activity}(e)$ (el nombre de la actividad que se está ejecutando) y $e = \#_{resource}(e)$ (el recurso que dispara el evento).

Cada línea en el *dotted chart* de la figura 87 se refiere a una clase, por ejemplo si se usa el clasificador $e = \#_{resource}(e)$ entonces cada línea corresponde a un recurso. Los puntos en esa línea describen eventos que perteneces a esa clase, por ejemplo, todos los eventos que se ejecutaron por un recurso particular. La dimensión tiempo puede ser absoluta o relativa. Si el tiempo es relativo, el primer evento de cada caso se ejecuta en tiempo cero. Por lo tanto la posición horizontal de los puntos dependen del tiempo que pasó desde que el primer evento para el mismo caso. La dimensión tiempo puede ser real o lógica. Para tiempo real, se usa el timestamp actual. Para el tiempo lógico, los eventos son simplemente enumerados sin considerar el timestamp actual: solo el orden se toma en cuenta. El primer evento tiene tiempo 0, el segundo evento tiene tiempo 1, etc. También el tiempo lógico puede ser absoluto (numeración global) o relativa (cada caso comienza en tiempo 0).

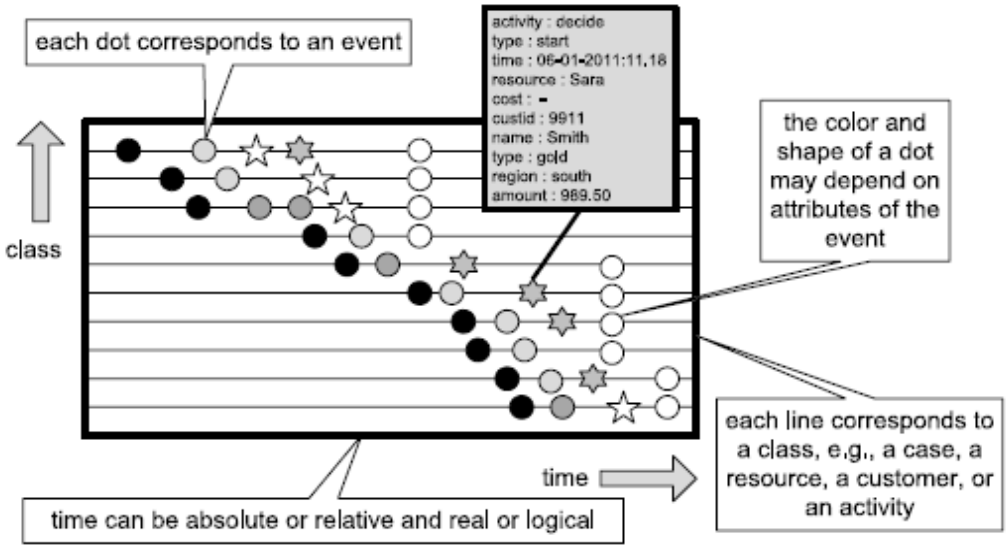


Figura 87 *Dotted chart*: los eventos se visualizan como puntos. La posición, el color y la forma dependen de los atributos del evento correspondiente. [4]

El color de un punto va a depender de otros atributos, es decir, también hay un clasificador para la forma del punto y un clasificador para el color de un punto. Por ejemplo, si se usa el clasificador es $e = \#_{case}(e)$, entonces cada línea corresponderá a un caso. La forma del punto puede cambiar dependiendo del recuso que dispara a el evento especifico y el color del punto puede depender del nombre de la actividad correspondiente.

El *dotted chart* es una herramienta poderosa para ver procesos desde diferentes ángulos. Se pueden ver todos los eventos e una mirada y mientras tanto mostrar diferentes perspectivas al mismo tiempo (clase, color, forma y tiempo). Además, haciendo zoom en una se pueden investigar patrones particulares. Por ejemplo cuando se use un clasificador $e = \#_{resource}(e)$ se puede ver cuando un recurso estuvo inactivo por mucho tiempo.

Es posible alinear a los eventos basándose en su contexto en vez de en el tiempo. Como resultado se alinean patrones repetitivos en el log de eventos, por eso es fácil ver comportamiento común y desviaciones sin necesidad de construir un modelo de proceso. La identificación de estos patrones ayuda a entender el comportamiento crudo que se captura en el log de eventos. Los logs de eventos suelen contener eventos de bajo nivel de poco interés para la administración. El desafío es agregar los eventos de bajo nivel en

eventos que tengan significado para los stakeholders. Por lo tanto el log de eventos por lo general se pre procesa luego de una inspección visual del log utilizando *dotted chart*. Hay varios enfoques para pre procesar logs de eventos de bajo nivel. Por ejemplo, por lo general la aparición de patrones de bajo nivel se puede abstraer en eventos que representan actividades a nivel de negocio.

El *dotted chart* se puede ver como un ejemplo de una técnica de análisis visual. La analítica visual aprovecha las capacidades humanas de identificar patrones y tendencias visualmente en conjuntos de datos grandes. [4]

A.3.8.1 Análisis de redes de trabajo social

La sociometría, también conocida como sociografía, se refiere a métodos que presentan datos en relaciones interpersonales en forma de grafo o de matriz.

Para las técnicas de *Process Mining* utilizaremos redes sociales de trabajo como los que se ven en la figura 88. Los nodos en una red de trabajo corresponden a entidades organizacionales. Muchas veces hay una correspondencia de uno a uno entre los recursos encontrados en el log y las entidades organizacionales (es decir nodos). Los nodos en una red de trabajo social pueden corresponder a entidades organizacionales agregadas como roles, grupos, y departamentos. Los arcos en una red de trabajo social corresponden a relaciones entre estas entidades organizacionales. Los arcos y nodos pueden tener pesos. El peso de un arco o de un nodo indica su importancia. La interpretación de la importancia depende de la red de trabajo social. Muchas veces se asocia la distancia para referirse a la inversa de un peso, un arco con un peso muy alto indica que la distancia es corta.

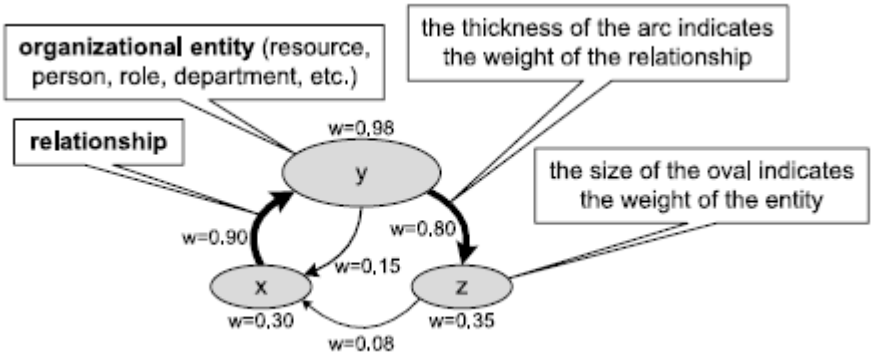


Figura 88 Un red de trabajo social consiste de nodos representando las entidades organizacionales, y arcos representando las relaciones. Tanto los nodos como los arcos pueden tener pesos indicados por "*w*=.." y el tamaño de la forma. [4]

Hay una gran variedad de métricas para analizar la red de trabajo social: centralización, cercanía, e intermediación. Nociones como centralización analizan la posición de una entidad organizacional, o persona en toda la red de trabajo social. El *índice de Bavelas–Leavitt* de centralidad es un ejemplo bien definido que se basa en las caminos geodésicos del grafo (es decir los caminos más cortos del grafo). Sea *i* el nodo y *Djk* la distancia geodésica desde el nodo *j* al nodo *k* el índice de centralidad *Bavelas–Leavitt* se define como $BL(i) = (\_{j,k}\ D_{j,k})/(\_{j,k}\ D_{j,i} + D_{i,k})$. El índice divide entre la suma de todas las distancias geodésicas por la suma de todas las distancias geodésicas desde y hace el nodo *i*. Otras

métricas relacionadas son la proximidad (1 dividido por la suma de todas las distancias geodésicas de un nodo dado) y la intermediación (una proporción basada en el número de caminos geodésicos visitando un nodo dado). Recordar que la distancia se puede ver como la inversa del peso de los arcos.

También hay métricas para hacer declaraciones sobre la red de trabajo como un todo como ser el grado de conectitud. Además hay también técnicas para identificar cliques (grupos de entidades que se encuentran fuertemente conectados unos con otros y que tienen menos conexiones con entidades fuera de la camarilla).
El log de eventos con el atributo #resource(e) proveen una fuente excelente de información para análisis de redes de trabajo. Por ejemplo, basándose en el log de eventos se puede contar el número de veces que se envía un trabajo de un recurso al otro.

Tabla 14 Matriz de trabajo mostrando los números significativos de pasaje de trabajo de una persona a otra por caso [4]

	Pete	Mike	Ellen	Sue	Sean	Sara
Pete	0.135	0.225	0.09	0.06	0.09	1.035
Mike	0.225	0.375	0.15	0.1	0.15	1.725
Ellen	0.09	0.15	0.06	0.04	0.06	0.69
Sue	0	0	0	0	0	0.46
Sean	0	0	0	0	0	0.69
Sara	0.885	1.475	0.59	0.26	0.39	1.3

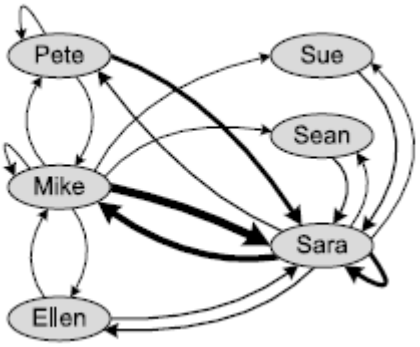


Figura 89 Red de trabajo social basada en el traspaso de trabajo a nivel individual de recursos utilizando el límite de 0.1. La estrechez de los arcos se basa en la frecuencia del paso de trabajo de una persona a otra. [4]

La tabla 5 muestra el número promedio de envíos de un recurso a otro. Todas las celdas que no sean 0 representan relaciones de entrega de trabajo. Cuando se visualiza una red de trabajo social, por lo general se pone un límite. Si seteamos como límite 0.1 obtendremos la red de trabajo social de la figura 89. Todas las celdas que tienen un valor de al menos 0.1 son convertidas en arcos en la red de trabajo. También se ve que hay una fuerte conexión entre Mike y Sara. Si aplicamos el índice de centralidad de Bavelas–Leavitt nos mostrará que Sara y Mike son los más centrales en la red de trabajo.

Los nodos en una red de trabajo social corresponden a entidades organizacionales. En la figura 89, las entidades son recursos individuales. Sin embargo, es posible construir

redes de trabajo social al nivel de departamentos, equipos, o roles. Si asumimos por ejemplo que hay tres roles: Asistente, experto y manager. Pete, Mike y Ellen tienen el rol de asistente. Sue y Sean tienen el rol de experto, y Sara es la única que tiene rol de administrador. Ahora podremos contar el número de entregas a nivel de rol. En la tabla 6 se muestra la frecuencia promedio de las entregas por caso. Esta matriz se puede convertir en la red de trabajo que se muestra en la figura 90. [4]

Tabla 15 Matriz de traspaso de trabajo a nivel de rol [4]

	Assistant	Expert	Manager
Assistant	1.5	0.5	3.45
Expert	0	0	1.15
Manager	2.95	0.65	1.3

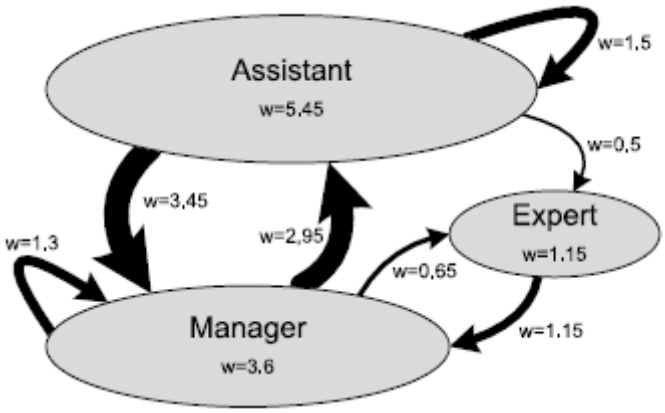


Figura 90 Red de trabajo social basada en el traspaso de trabajo a nivel de rol. El peso de los nodos se basa en la cantidad de veces que un recurso de determinado rol realiza una actividad. Los pesos de los arcos se basan en el número promedio de veces que se pasa trabajo de un rol a otro por caso. [4]

Descubrimiento de estructuras organizacionales

El comportamiento de un recurso se puede caracterizar por un perfil es decir, un vector indicando cada cuanta cada actividad ha sido ejecutado por un recurso. Con estos perfiles, se pueden emplear técnicas de clustering que se utilizan para descubrir similitud entre recursos. En la figura 90 se ve un ejemplo en el que se descubren tres roles basándose en las similitudes de los perfiles de los seis recursos. Luego de clusterizar los recursos en grupos, estos grupos se pueden relacionar a actividades en el proceso como se ve en la figura 92

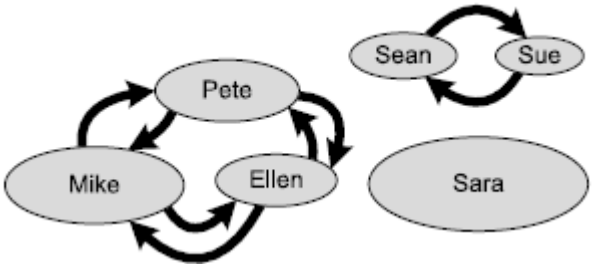


Figura 91 Red de trabajo social basada en las similitudes de los perfiles. Los recursos que ejecutan una colección similar de actividades se relacionan. Sara es el único recurso que ejecuta *e* y *f* por lo tanto no se conecta con los otros recursos. Las vueltas a uno mismo se descartan ya que no contienen información. [4]

.

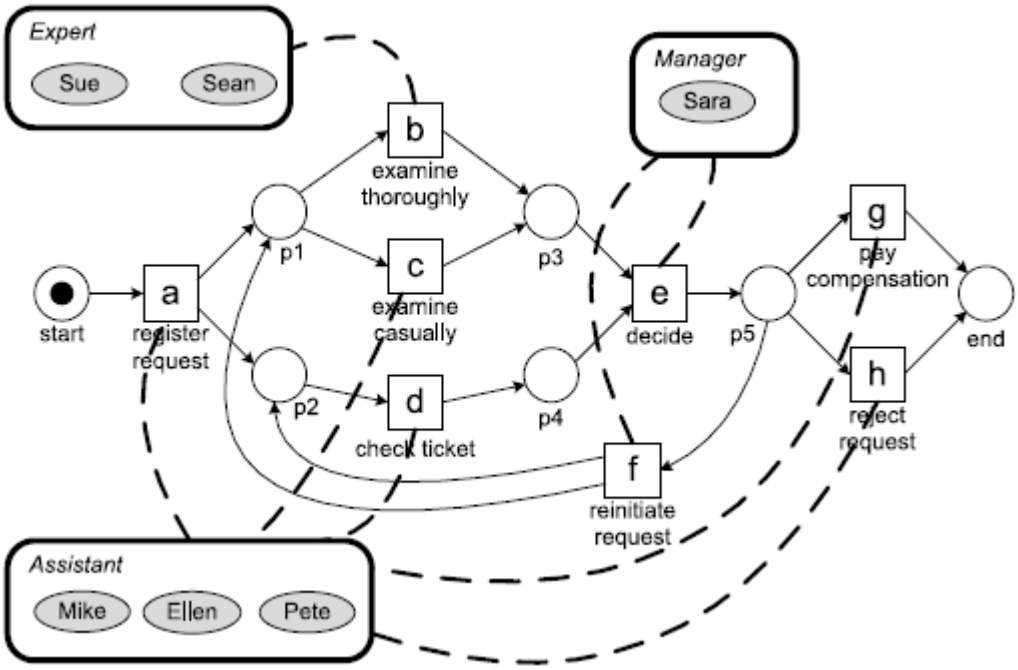


Figura 92 Modelo organizacional descubierto basándose en el log de eventos. [4]

Los tres roles Asistente, experto y manager en la figura 91 tienen la propiedad de que particionan el conjunto de recursos. Por lo general este no es el caso, por ejemplo un recurso puede tener múltiples roles (ejemplo un consultor que también sea el líder del equipo). Además cada actividad corresponde a un rol precisamente. Esto tampoco tiene que ser el caso siempre. La figura 92 muestra una situación más general.

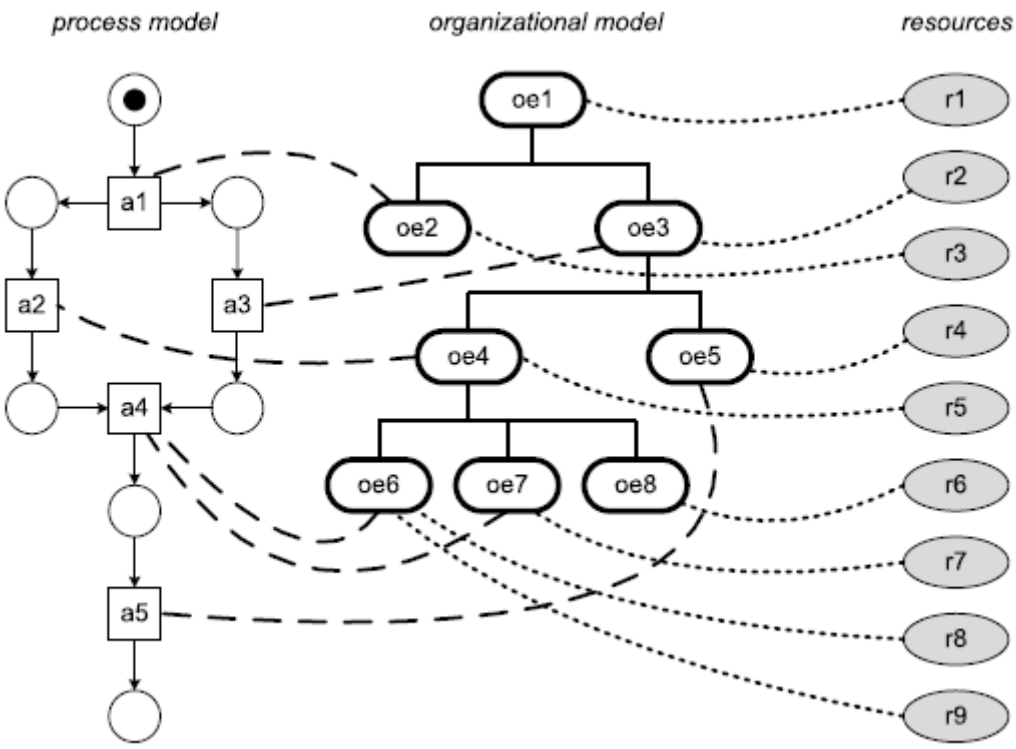


Figura 93 Las entidades organizacionales descubiertas conectan actividades en el modelo de procesos con conjuntos de recursos. [4]

En la figura 93 se conecta al modelo de proceso y los recursos vistos en el log de eventos. Hay ocho entidades organizacionales. El modelo es jerárquico.

Anexo 4 Implementación del proceso de Solicitud de compra

En este anexo se detalla la implementación del proceso de solicitud de compra y los subprocesos solicitud de cotizaciones y orden de compra. Comenzaremos por el proceso padre y luego continuaremos con los subprocesos, por cada uno de ellos se verán las siguientes etapas de implementación:

- Declaración de variables globales de proceso, que puede ser de los tipos prefinidos o puede ser un tipo Java complejo.
- Declaración de variables de tarea: que puede ser de los tipos prefinidos o puede ser un tipo Java complejo.
- Diseño de formularios o implementación de tarea automática: en este paso se diseña cada uno de los formularios de las tareas manuales o se codifican las tareas automáticas. En el diseño de formularios se debe conectar cada uno de los widgets de los mismos con las variables de tarea o proceso y se permite agregar código java para procesar la visualización de cada uno de los widgets con código script Java

A.4.1 Solicitud de compra

Las variables del proceso de *solicitud de compra* son las siguientes:

- *Costo*
- *Departamento*
- *Dirección de entrega*
- *Fecha de entrega*
- *IdCotizacion*
- *idSolicitud*
- *cotización elegida*
- *solicitud correcta*
- *solicitud aporbada*
- *solicitador*
- *puntaje calidad*

En el proceso principal de solicitud de compra en la primer tarea “Crear solicitud de compra” se posee un conector, que se ejecuta al inicio de la misma, que obtiene los productos de la base de datos, para luego desplegarlos en el formulario para que el empleado operario puedo seleccionar los solicitados. El conector es un conector para Mysql que ejecuta la siguiente consulta: “`select nombre, precioUnitario from producto`”. Luego la salida de la consulta se procesa y se almacena en la variable del proceso *productos* que es un ArrayList con dos ArrayList, uno con los nombres de los productos y otro con los precios de los productos. Luego esa variable es utilizada como entrada en la matriz editable del formulario, el formulario de esa tarea se ve en la figura 94.

Figura 94 Diagrama base de datos del departamento de compras

Figura 95 Formulario de la tarea "Crear solicitud de compra"

En la siguiente tarea "Visualizar solicitud de compra" el empleado puede visualizar la solicitud de compra generada y en caso de ser necesario modificarla volviendo nuevamente a la tarea "Crear solicitud de compra". El formulario de esta tarea se despliega en la figura 96.

Figura 96 Formulario de la tarea "Visualizar solicitud compra"

La siguiente tarea "Autorizar solicitud", debe ser llevada a cabo por el jefe de empleados, en esta instancia el formulario es como el que se muestra en la figura 97, donde se visualizan los campos de la solicitud de compra y le permite al usuario aprobar o desaprobar la solicitud. Al inicio de la tarea se poseen tres conectores con la base de datos del departamento de compras:

- Insertar solicitud de compra: inserta una tupla en la tabla solicitud compra el código del conector es el siguiente:

```
java.text.SimpleDateFormat sdf = new
java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
String fecha= sdf.format(fecha_entrega);

return "INSERT INTO solicitudcompra (nombreUsuario,
justificacionCompra, departamento, fechaEntrega,
direccionEntrega) VALUES ('"+solicitador+"' ,
'"+justificacion+"' , '"+departamento+"' , '"+fecha+"' ,
'"+direccionEntrega+"')";
```

  En la primer parte se aplica el formato a la fecha para ser almacenada en un formato legible por la herramienta ProM. Luego se genera la consulta SQL y se le pasan las variables correspondientes

- Obtener id solicitud: como el campo idSolicitud es una clave autoincremental, se realiza una consulta para obtener el id de la ultima solicitud agregada:

```
SELECT MAX(idSolicitudCompra) FROM solicitudCompra
```

  Luego el resultado de la consulta se almacena en la variable idSolicitud
- Insertar productos-solicitud de compra: En este paso se insertan las tuplas correspondientes en la tabla *solicitudProducto:*

```
String result= "";

for(List<String> producto : productosSolicitud){
     result += "INSERT INTO solicitudProducto
(`idSolicitud`, `idProducto`,  `cantidad`) "
     String nombreProd = producto.get(0);
     String cantProd = producto.get(2);
     result += "VALUES ( '"+idSolicitud+"', (SELECT
idProducto FROM producto WHERE nombre =
'"+nombreProd+"'), "+cantProd+" );";
}
return result;
```

Actualizar estado solicitud, una vez que se completa la tarea, se ejecuta el conector que actualiza el campo aprobado de la tabla solicitudCompra, el código del conector es el siguiente:

```
return "UPDATE solicitudcompra SET
aprobado="+solicitudAprobada+" WHERE
idSolicitudCompra='"+idSolicitud+"'";
```

Figura 97 Formulario de la tarea "Autorizar solicitud"

Si la solicitud de compra fue rechazada de ejecuta la tarea "Notificar rechazo" que notifica al empleado operario del rechazo de la solicitud de compra, en la figura 98 se ve un formulario de la tarea "Notificar rechazo".

Si la solicitud de compra es aprobada se ejecuta el subproceso de cotización, a este subproceso se le pasan como parámetro las siguientes variables: idsolicitud y productosSolicitud y devuelve la variable cotización elegida. Al finalizar el mismo se ejecutan tres conectores que se encargan de setear las variables: idCotizacion,

puntaje_calidad y costo obteniendo los valores de la variable cotización elegida devuelta por el subproceso.

Luego se ejecuta el subproceso orden de compra al cual se le pasan como parámetro las siguientes variables: idCotizacion, fecha_entrega, costo, puntaje_calidad, solicitador, departamento, productosSolicitud.

A continuación entraremos en detalle en la implementación del subproceso de cotización, y el subproceso de orden de compra.

Figura 98 Formulario de la tarea notificar rechazo

A.4.2 Cotización

En este proceso se gestionan las cotizaciones para la solicitud de compra generada. Este proceso es llevado a cabo por un empleado administrativo.

El proceso tiene las siguientes variables de proceso:

- *Cantidad de proveedores*
- *Cotización elegida*
- *Cotizaciones*
- *IdSolicitud,* la obtiene del subproceso padre
- *Lista de proveedores*
- *Productos,* la obtiene del subproceso padre.
- *Proveedor a cotizar*
- *Proveedores,* contiene a todos los proveedores de los productos seleccionados.
- *Proveedores seleccionados,* contiene a los proveedores seleccionados pedir cotización.
- *idCotizacion*

La primer tarea es la de "Seleccionar proveedores" en esta tarea se visualiza la lista de proveedores disponibles para los productos solicitados y se deben seleccionar a los que se le quiere pedir una cotización. Al iniciar la tarea se ejecuta un conector que obtiene los proveedores de los productos, el código del conector es el siguiente:

```
String result="SELECT DISTINCT nombreProveedor,
telefono, correoElectronico FROM proveedor INNER JOIN
productoProveedor ON(proveedor.idProveedor =
productoProveedor.idProveedor) INNER JOIN producto ON
(productoProveedor.idProducto= producto.idProducto)";
result +="WHERE ";
```

```
        for(ArrayList<String> producto : productos){
             result +="producto.nombre = '"+producto.get(0)+"'";
             if(productos.indexOf(producto) < productos.size()-
1){
                  result +=" OR ";
             }
        }
        return result;
```

Luego se despliega el formulario como el que se ve en la figura 99.

Figura 99 Formulario actividad "Seleccionar proveedores".

Una vez seleccionados los proveedores se les notifica los productos que se quieren comprar y se espera la cotizacion de los mismos. Cuando el proveedor se ponga en contacto para cotizar el producto, el empleado administrativo debe ejecutar la tarea "Recibir cotizaciones de proveedor", esta tarea posee un formulario como el que se muestra en la figura 100. En este formulario se debe seleccionar el proveedor que nos va a transmitir la cotizacion.

Figura 100 Formulario actividad "Recibir cotizaciones de proveedor"

Figura 101 Formulario tarea "Completar cotización"

La siguiente tarea, “Completar cotización” posee un formulario como el que se ve en la figura 101. Al finalizar la tarea se ejecutan seis conectores:

- Decrementar proveedores: decrementa la variable cantidad de proveedores donde se almacena la cantidad de proveedores que deben presentar cotización
- Quitar proveedores: se quita de la lista de proveedores a cotizar, al proveedor que a continuación comunicara la cotización.
- Insertar cotización: se inserta en la tabla cotización los datos de la nueva cotización. El código del conector es el siguiente:

```
     java.text.SimpleDateFormat sdf = new
java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
     String fecha_e= sdf.format(fecha_entrega);
     String result= "INSERT INTO cotizacion
(idSolicitud, fechaEntrega, costoTotal, descuento,
puntajeCalidad, nombreProveedor ) VALUES ( "+idSolicitud
+", '"+ fecha_e+ "', "+costo_total+", "+descuento+",
'"+puntaje_calidad+"', '"+proveedor_a_cotizar+"')";
return result;
```

- Obtener idCotizacion, se obtiene el id de la cotizacion insertada y se almacena en la variable idCotización, el código del conector es el siguiente:

```
     SELECT MAX(idCotizacion) FROM cotizacion
```

- Insertar productos cotización: inserta en la tabla productoCotizacion

```
     String result= "";
     int i=0;
     for(List<String> producto : cotizacion_productos){
          result += "INSERT INTO productoCotizacion
(`idCotizacion`, `idProducto`, `nombreProducto`,
`cantidad`, `precioUnitario`, `precioTotal` ) "
          ArrayList temp = productos.get(i);
          String nombreProd = temp.get(0);
          String cantProd = producto.get(0);
          String precioUnitario= producto.get(1);
          String precioTotal= producto.get(2);
          result += "VALUES ( '"+idCotizacion+"',
(SELECT idProducto FROM producto WHERE nombre =
'"+nombreProd+"'), '"+nombreProd+"', "+cantProd+",
"+precioUnitario+", "+precioTotal+" );";
          i++;
     }
     return result;
```

- Insertar cotización proveedor se inserta una tupla en la tabla cotizacionproveedor, el código del conector es el siguiente:

```
     return "INSERT INTO cotizacionproveedor
(`idCotizacion`, `idProveedor`)
```

```
VALUES('"+idCotizacion+"',(SELECT idProveedor FROM
proveedor WHERE
nombreProveedor='"+proveedor_a_cotizar+"'))";
```

Una vez que se completa la cotizacion, si todavía no cotizaron todos los proveedores se vuelve a la tarea "Recibir cotizaciones de proveedor" si ya se recibieron todas las cotizaciones se prosigue con la ultima tarea del proceso: "Seleccionar cotización".

En la tarea "Seleccionar cotización" se ejecutan tres conectores:

- Seleccionar cotizaciones de solicitud: se consulta en la base de datos todas las cotizaciones realizadas por los proveedores, el código del conector es el siguiente:

```
return"SELECT idCotizacion, nombreProveedor,
descuento, fechaEntrega, puntajeCalidad, costoTotal FROM
cotizacion WHERE idSolicitud="+idSolicitud;
```

- Establecer cotización elegida: se establece la variable cotización elegida.
- Actualizar el estado de la cotizacion elegida: se actualiza en la base de datos el estado de la cotizacion elegida, el código del conector es el siguiente:

```
return
"UPDATE cotizacion SET aprobado=1 WHERE
idCotizacion='"+cotizacion_elegida.get(0)+"'";
```

El formulario de la tarea "seleccionar cotización" es como el que se muestra en la figura 102, se despliegan las cotizaciones para la solicitud y el empleado administrativo debe elegir una.

Figura 102 Formulario tarea "Seleccionar cotización"

A.4.3 Orden de compra

Una vez que se ejecuta el subproceso de cotización, se ejecuta el proceso de orden de compra. Las variables del proceso de orden de compra son las siguientes:

- *Aprobado*
- *Costo total*, la recibe como parámetro del proceso padre
- *Departamento,* la recibe como parámetro del proceso padre
- *Fecha entrega*, la recibe como parámetro del proceso padre
- *Fecha orden*
- *idCotizacion*, la recibe como parámetro del proceso padre
- *Numero de orden*
- *Opción de pago*
- *Productos,* la recibe como parámetro del proceso padre
- *Puntaje calidad*, la recibe como parámetro del proceso padre
- *Requiere aprobación*
- *Solicitador*, la recibe como parámetro del proceso padre

En la primer tarea "Crear orden de compra" se despliega un formulario como el que se ve en la figura 103. En este formulario se completan los datos de la orden de compra, y se debe indicar si la misma requiere aprobación.

Figura 103 Formulario actividad "Crear orden de compra"

Si la tarea requiere aprobación se ejecuta la tarea "Aprobar orden de compra" que es ejecutada por un gerente administrativo, y posee un formulario como el de la figura 104, donde el gerente administrativo debe indicar si aprueba o no la orden de compra.

Figura 104 Formulario tarea "Aprobar orden de compra"

Si la orden de compra fue aprobada se ejecuta la tarea "Visualizar orden de compra" que posee un formulario como el de la figura 105. Al finalizar el formulario se ejecuta un conector que inserta la orden de compra aprobada en la base de datos. El código del conector es el siguiente:

```
     java.text.SimpleDateFormat sdf = new
java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
     String fecha_e= sdf.format(fecha_entrega);
     String fecha_o= sdf.format(fecha_orden);
     return "INSERT INTO ordendecompra (idCotizacion,
puntajeCalidad, costo, fechaEntrega, fechaOrden,
numeroDeOrden, opcionDePago, aprobado) VALUES
("+idCotizacion+", '"+puntaje_calidad+"', "+costo_total+" ,
'"+fecha_e+"', '"+fecha_o+"', "+numero_de_orden+",
'"+opcion_de_pago+"', "+aprobado+" )";
```

Figura 105 Formulario tarea "Visualizar orden de compra"

Si la orden de compra fue rechazada se ejecuta la tarea "Notificar rechazo" que inserta en la tabla ordedecompra la orden de compra rechazada. La tarea "Notificar rechazo" tiene un formulario como el que se ve en la figura 106.

Figura 106 Formulario tarea "Notificar rechazo"

Anexo 5 Aplicación de las técnicas de *Process Mining* al proceso de solicitud de compra

En este anexo se detalla el resultado de la aplicación de las técnicas de *Process Mining* sobre el proceso de solicitud de compra, mostrando los resultados del análisis. Las técnicas de *Process Mining* a analizar son *View Inspector*, *dotted chart*, *BPMN analysis*, sistema de transición y *Fuzzy miner*

A.5.1 View Inspector

A.5.1.1 Solicitud de compra

Al importar el log logSolicitudDeCompra.xes ProM nos brinda la posibilidad de visualizar el log de eventos con el *View Inspector* para obtener una visión general del mismo. En la figura 107 se puede ver los diagramas de los eventos por caso, y abajo las clases de eventos por casos también. Como datos clave nos dice la cantidad de procesos analizados, la cantidad de casos y la cantidad total de eventos que se observan en el log. Luego en otro grupo la cantidad de clases de eventos y la cantidad de tipos de eventos y finalmente la cantidad de originadores.

Los valores obtenidos como se ven en la figura son:

- Procesos 1: se analiza únicamente el proceso de solicitud de compra
- Casos 22: Se registraron 22 instancias del proceso solicitud de compra.
- Eventos: 252 registrados para el proceso de solicitud de compra
- Clases de eventos: hay 14 clases de eventos distintas en el proceso de solicitud de compra, conformados por las eventos de inicio y de fin las compuertas y las distintas actividades (manuales/ automáticas) que conforman al proceso de solicitud de compra.

- Hay cuatro originadores que pueden iniciar el proceso.

Luego también poseemos información del log con la fecha de inicio, del primer caso registrado y la fecha de fin del último caso registrado.

Figura 107 *View Inspector* aplicado al proceso de solicitud de compra

En la pestaña *Inspector* del *View Inspector* se puede ver el detalle de todas las instancias de proceso y de cada uno de los eventos que la componen y los atributos tanto de las instancias de proceso como de los eventos. Figura 108.

Dentro del *Inspector* en la pestaña explorer se puede ver las instancias de proceso registradas en el log con grafico de los eventos de cada una por color. Si es verde son los eventos más frecuentes y colorados los menos frecuentes. Se puede deslizar el mouse por los eventos para ver más detalles de los mismos, fecha de ocurrencia del evento, participante que lo realizo y frecuencia de ocurrencia del mismo, en nuestro proceso de solicitud de compra las tareas que se encuentran naranjas son las de los subprocesos y la de notificar rechazo, que por lo general son las tareas que más demoraron (Figura 109).

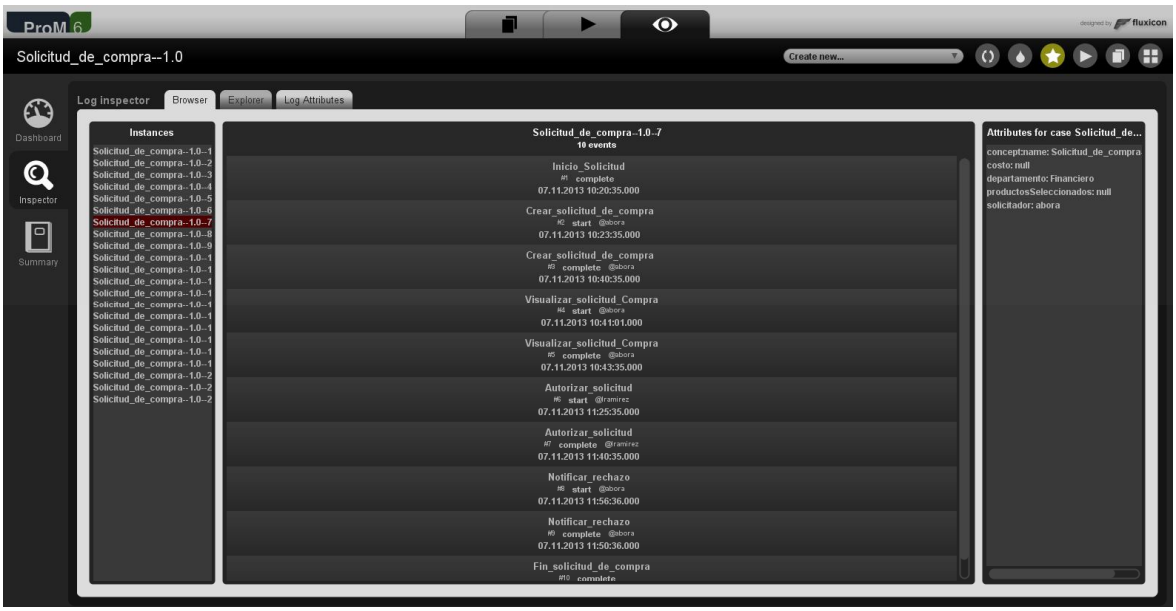


Figura 108 Browser aplicado al proceso de solicitud de compra

En la pestaña *log atributes* del *log inspector* se puede visualizar el detalle de los atributos de instancia y de evento como se ve en la figura 110.

Por ultimo en la pestaña *Summary* del *View Inspector* se poseen datos de análisis de proceso resumidos. Eventos de proceso registrado cantidad de ocurrencias de cada uno y frecuencia de ocurrencia de los mismos.

Eventos de inicio y de fin detectados.

Información de los participantes, frecuencia de participación y cantidad de veces que realizaron algún evento.

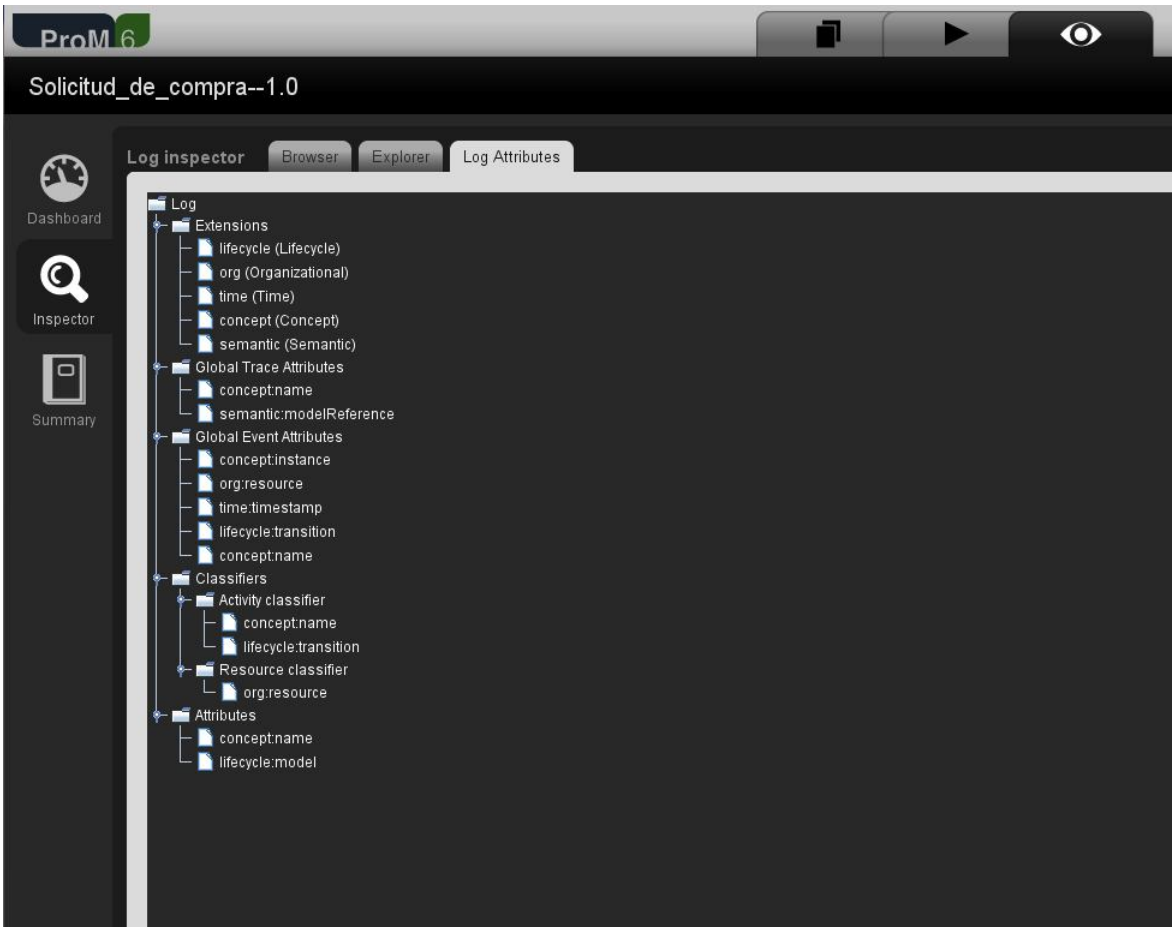


Figura 109 Log Attributes aplicado al proceso de solicitud de compra

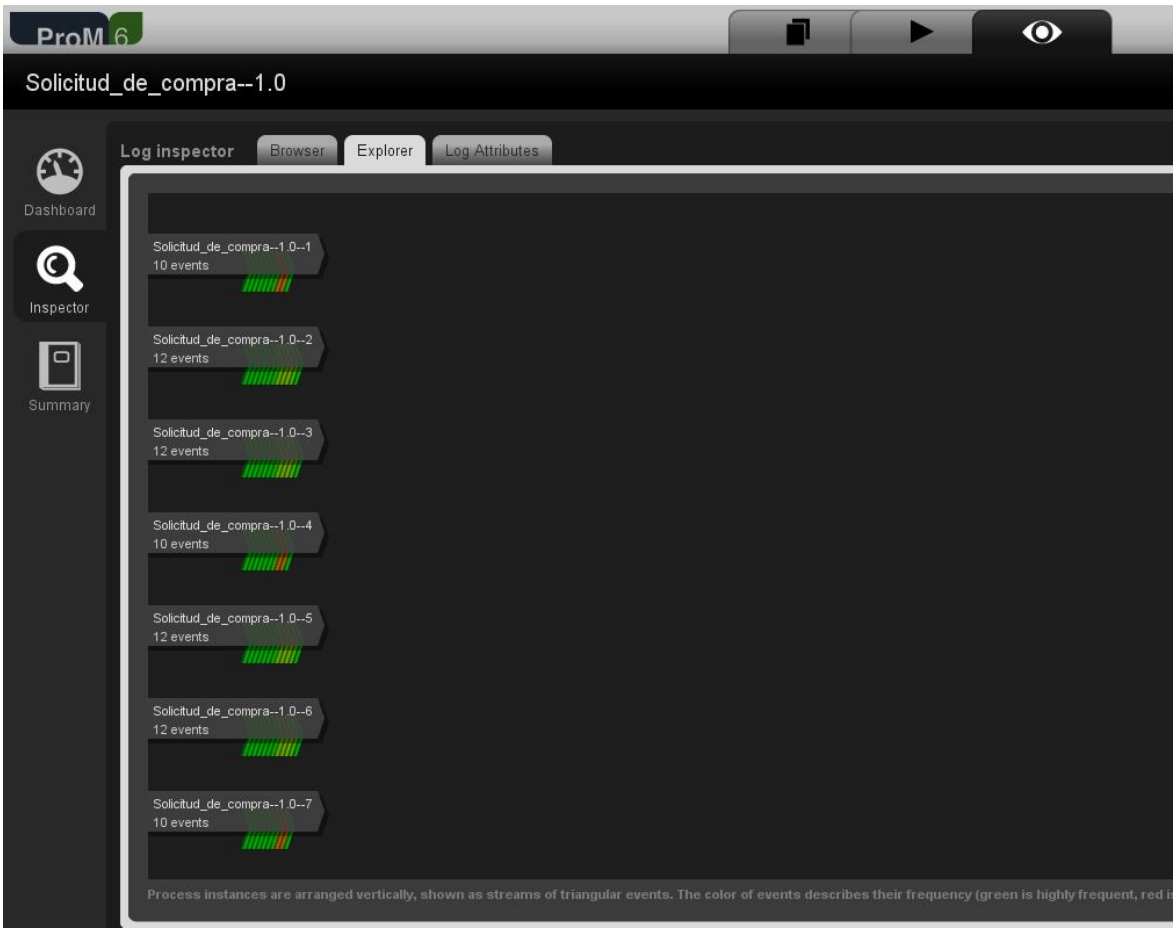


Figura 110 Explorer aplicado al proceso de solicitud de compra

Con el soporte de esta herramienta se puede importar el log de eventos y tener una visión general del proceso a analizar.

Contamos con tres logs de eventos el log logSolicitudDeCompra.xes que es el proceso padre y los subprocesos: logCotizacion.xes y logOrdenDeCompra.xes. En análisis del proceso logSolicitudDeCompra.xes se verán los subprocesos logCotizacion.xes y logOrdenDeCompra.xes como una tarea automática, son una caja negra donde solo se sabe información del inicio y del fin de los subprocesos para realizar un análisis profundo de cada uno de ellos se debe importar el subproceso deseado y realizar el correspondiente análisis del mismo.

A.5.1.2 Solicitud cotizaciones

Al importar el log de eventos logCotizaciones del subproceso *Solicitud cotizaciones* con el log inspector se puede observar que se comenzaron a registrar datos desde el viernes 1 de noviembre a las10:27 hs y que el ultimo evento registrado en el log fue el Lunes 18 de noviembre a las 9:57 hs.

Se obtiene que el proceso se ejecutó 15 veces, y que el log contiene 186 eventos para todas las instancias registradas. Hay diez clases de eventos distintas, 2 tipos de eventos distintos (start, complete) y tres participantes fbravo, alara y registra como participante nulo para aquellos eventos automáticos. Fbravo interactúa en 108 eventos mientras que alara en 48.

A.5.1.3 Orden de compra

Al importar el log de eventos logOrdenDeCompra del subproceso Orden de compra se puede observar con el log inspector que la fecha de inicio del log es el viernes 1 de noviembre a las 12:10 hs y que finaliza el Lunes 18 de noviembre a las 11:23 hs.

El subproceso se ejecutó 15 veces al igual que el subproceso de solicitud de cotizaciones ya que la ejecución de los mismos es secuencial. El subproceso tiene 104 eventos. Hay diez clases de eventos, dos tipos de eventos y cuatro participantes, fbravo, alara, plansky y registra como participante nulo para aquellos eventos automáticos. Plansky es el gerente administrativo y solo interactúa en aquellas instancias en las cuales la orden de compra requiere aprobación tiene una ocurrencia de un 13%. Fbravo y alara tienen el mismo rol y pueden ejecutar las mismas tareas pero sin embargo fbravo tiene 40 ocurrencias mientras que alara 20.

A.5.2 Dotted chart

Dotted chart es otra de las técnicas que permiten realizar un análisis sobre el procesos de negocio. Esta técnica permite visualizar información del proceso organizada sobre un gráfico de X e Y donde X es el tiempo y el eje Y se puede configurar para mostrar instancia de proceso, instancia de actividad, participante, o evento.

El *dotted chart* junto con el log inspector permiten realizar análisis sobre los procesos de negocio implantados en un BPMS. Permiten evaluar el proceso en cuanto a las instancias de proceso, las tareas del proceso, los participantes y la performance en general.

A.5.2.1 Solicitud de compra

Si aplicamos la técnica de *Dotted chart* sobre el proceso de solicitud de compra desplegando la instancia de proceso sobre el eje Y se visualizara un gráfico como el de la figura 111. Los puntos representan la ocurrencia de las instancias de proceso en el tiempo. A la derecha de la pantalla se puede visualizar un detalle por instancia de proceso mostrando el tiempo de inicio y de fin de cada instancia.

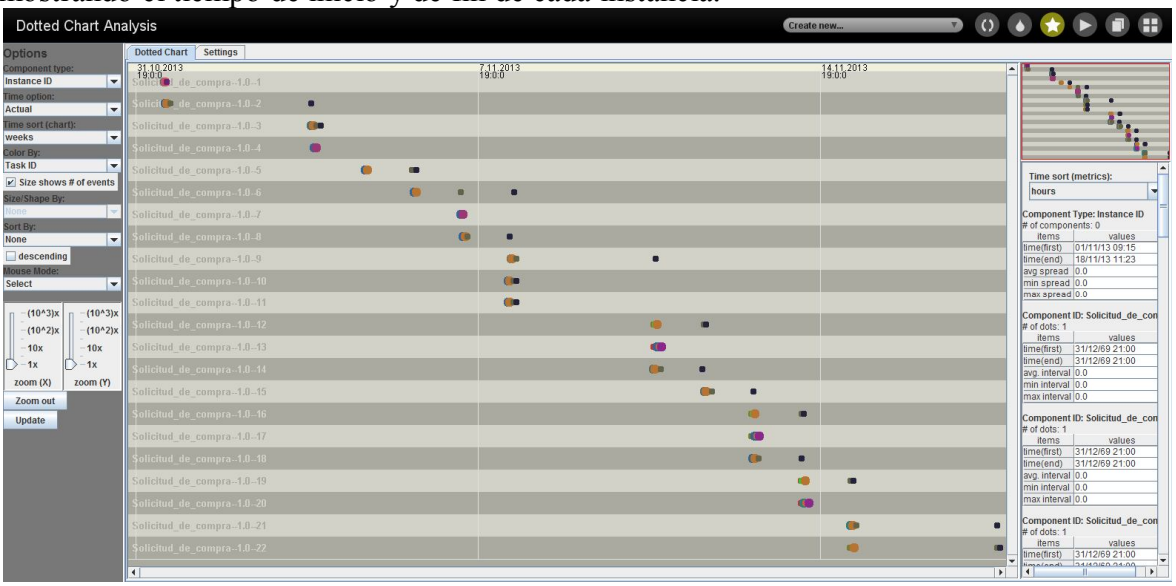


Figura 111 *Dotted chart* aplicado al proceso de solicitud de compra por instancia de proceso

Si aplicamos la técnica sobre el proceso de solicitud de compra desplegando el participante sobre el eje Y se visualizara un gráfico como el de la figura 112. Donde se puede ver la interacción de los participantes en cada tarea a lo largo del tiempo. Como se ve en la figura los participantes del proceso solicitud de compra son abora, mgerez, y lramirez. Siendo abora el participante con más interacciones en el proceso.

Si aplicamos la técnica sobre el proceso de solicitud de compra desplegando la tarea sobre el eje Y se visualizara un gráfico como el de la figura 113. Donde se puede ver como se ejecutaron las tareas a través del tiempo.

Como puede verse en las figuras 111, 112, 113 se ven días en los que no se registraron interacciones de ningún tipo, esto se debe a que fueron fin de semanas.

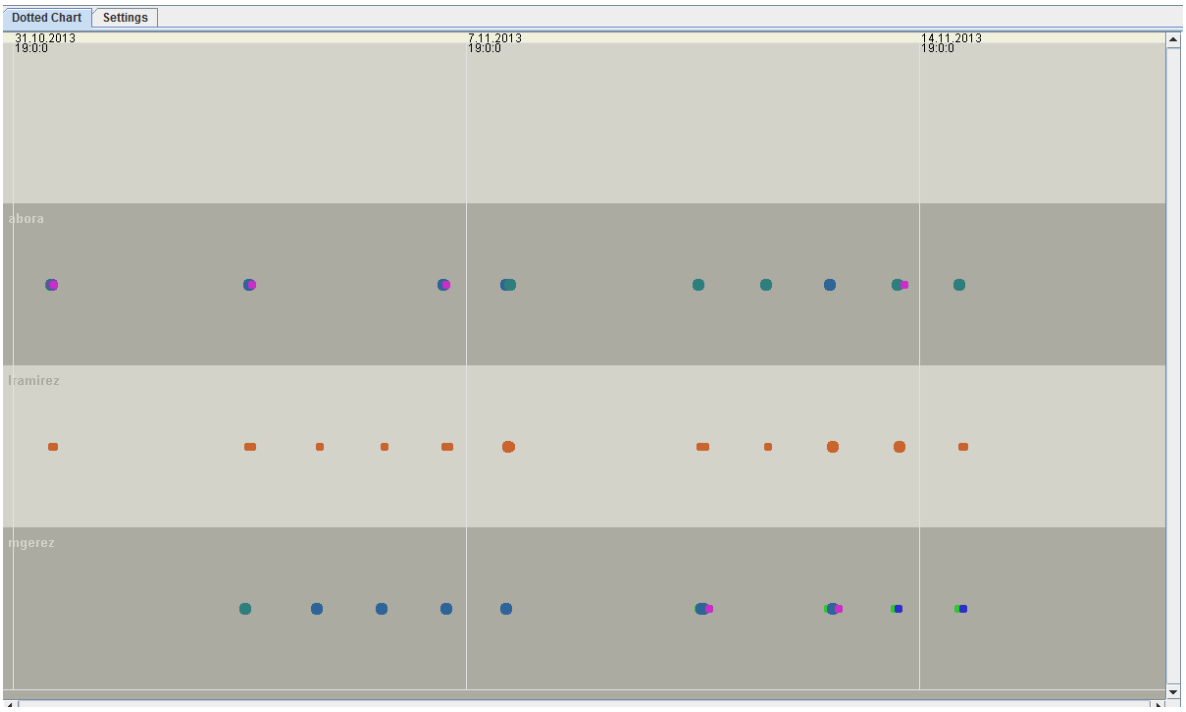


Figura 112 *Dotted chart* aplicado al proceso de solicitud de compra desplegado por participante

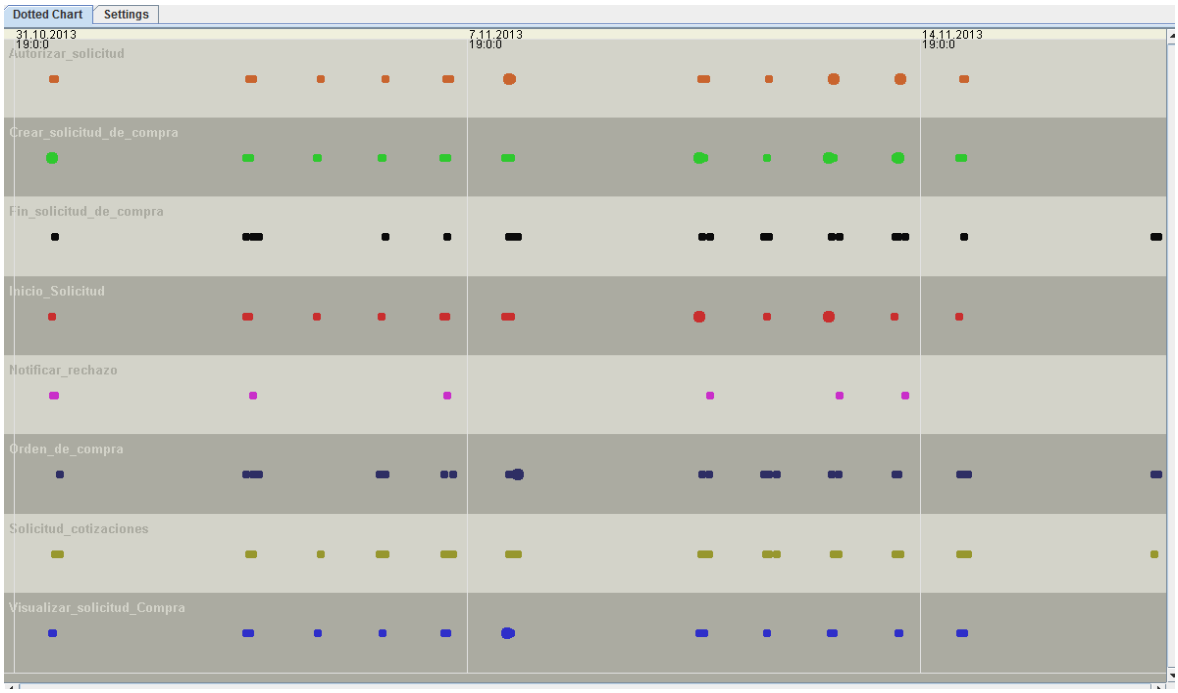


Figura 113 *Dotted chart* aplicado al proceso de solicitud de compra desplegado por tarea.

A.5.2.2 Solicitud cotizaciones

Si aplicamos la técnica de *dotted chart* al subproceso de solicitud de cotizaciones por instancia se obtiene un gráfico como el de la figura 114 donde cada punto representa la ocurrencia de la instancia en el tiempo.

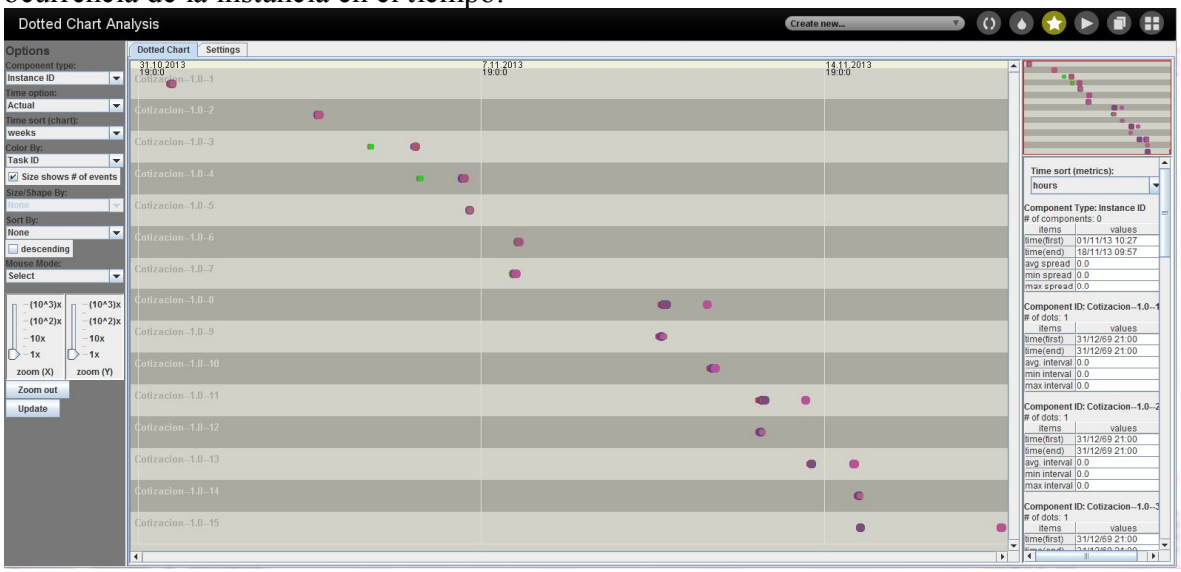


Figura 114 *Dotted chart* aplicado al proceso de solicitud de cotizaciones por instancia de proceso

Si aplicamos la técnica sobre el proceso de solicitud de cotizaciones desplegando el participante sobre el eje Y se visualizara un gráfico como el de la figura 115. Donde se puede ver la interacción de los participantes en cada tarea a lo largo del tiempo. Como se ve en la figura los participantes del proceso solicitud de compra son alara, fbravo y registra un

participante que realiza tareas automáticas. Siendo fbravo el participante con más interacciones en el proceso.

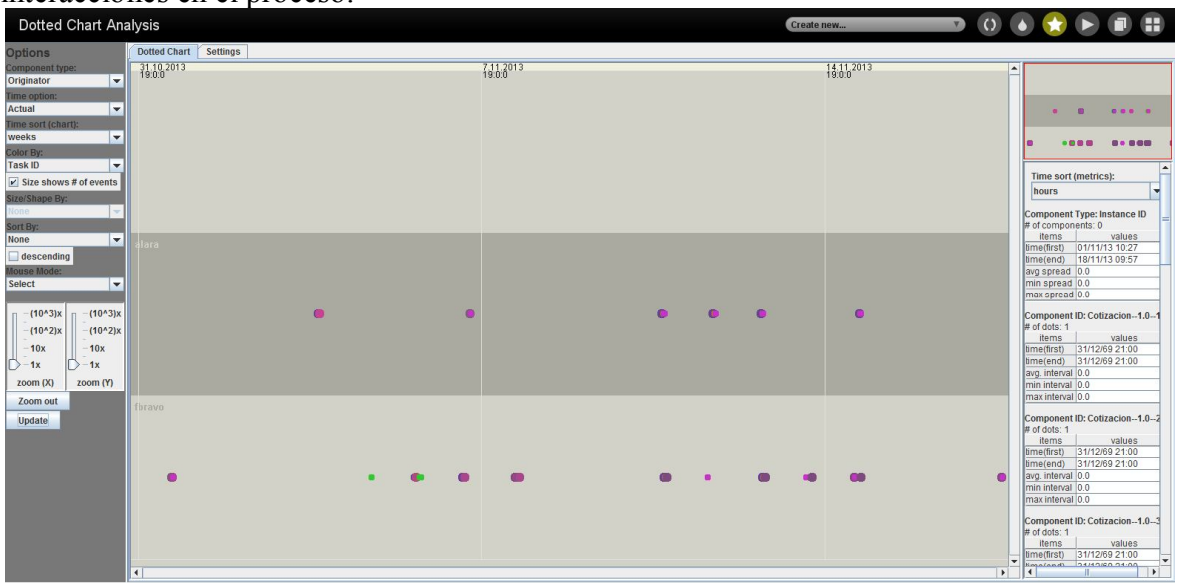


Figura 115 *Dotted chart* aplicado al proceso de solicitud de cotizaciones por participantes del proceso

Si aplicamos la técnica sobre el proceso de solicitud de cotizaciones desplegando la tarea sobre el eje Y se visualizara un gráfico como el de la figura 116. Donde se puede ver como se ejecutaron las tareas a través del tiempo.

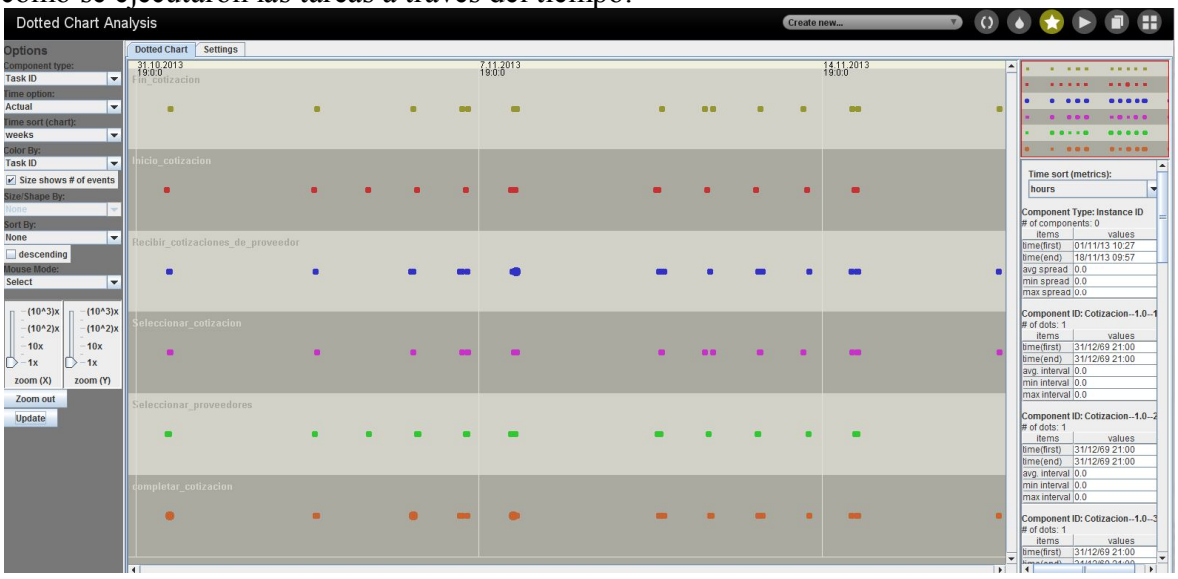


Figura 116 *Dotted chart* aplicado al proceso de solicitud de cotizaciones por tarea del proceso

Como puede verse en las figuras 114, 115, 116 se ven días en los que no se registraron interacciones de ningún tipo, esto se debe a que fueron fin de semanas.

A.5.2.3 Orden de compra

Si aplicamos la técnica de *dotted chart* al subproceso orden de compra por instancia se obtiene un gráfico como el de la figura 117 donde cada punto representa la ocurrencia de la instancia en el tiempo.

Figura 117 *Dotted chart* aplicado al proceso de orden de compra por instancia de proceso

Si aplicamos la técnica sobre el proceso de orden de compra desplegando el participante sobre el eje Y se visualizara un gráfico como el de la figura 118. Donde se puede ver la interacción de los participantes en cada tarea a lo largo del tiempo. Como se ve en la figura los participantes del proceso solicitud de compra son alara, fbravo y plansky registra un participante que realiza tareas automáticas. Se observa por el color de los puntos que plansky realiza tareas diferentes a las que realiza alara y fbravo, siendo alara y fbravo dos participantes con el mismo rol. Fbravo participa más en el proceso que alara.

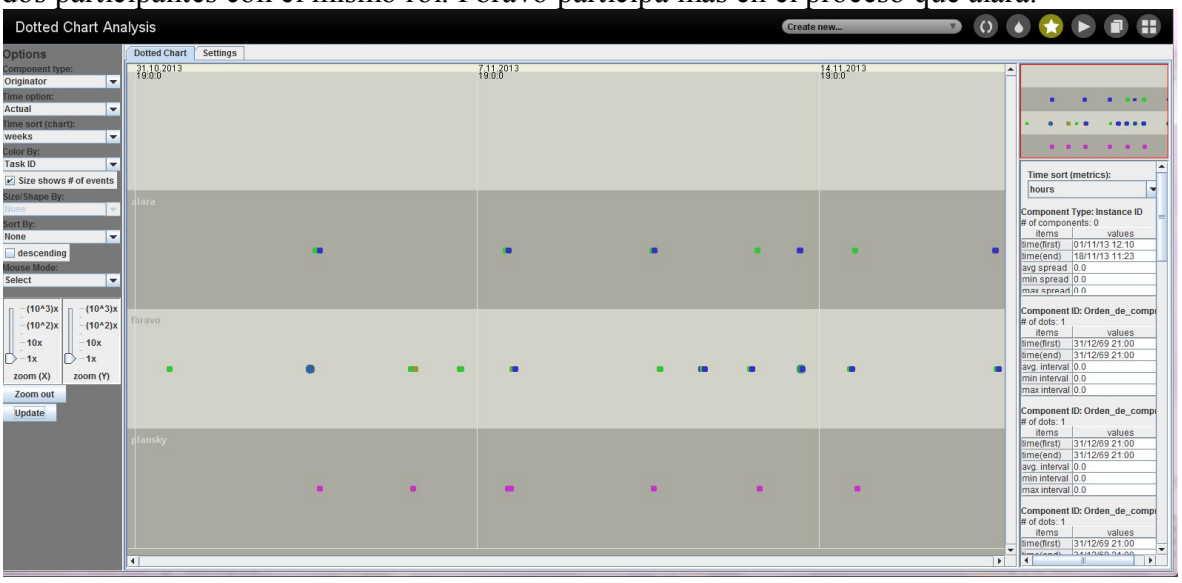


Figura 118 *Dotted chart* aplicado al proceso de orden de compra por participante de proceso

Si aplicamos la técnica sobre el proceso de orden de compra desplegando la tarea sobre el eje Y se visualizara un gráfico como el de la figura 119. Donde se puede ver como se ejecutaron las tareas a través del tiempo.

Figura 119 *Dotted chart* aplicado al proceso de orden de compra por tarea de proceso

Como puede verse en las figuras 117, 118, 119 se ven días en los que no se registraron interacciones de ningún tipo, esto se debe a que fueron fin de semanas.

A.5.3 BPMN analysis

La aplicación de la técnica de análisis BPMN permitirá visualizar el modelo en notación BPMN y nos permitirá hacer análisis de performance. También permite analizar las interacciones entre los participantes por medio de redes de trabajo social. Si se compara con el proceso obtenido del análisis con el modelado en Bonita Open Solution se puede ver que las actividades, flujos y compuertas se corresponden. Se puede observar también que los arcos varían su grosor, cuanto más fino sea el arco menos veces se toma ese camino mientras que cuanto más grueso sea más frecuencia de ocurrencia tiene ese camino.

BPMN analysis permite identificar un número determinado de caminos más frecuentes, por defecto identifica tres caminos más frecuentes y permite analizar y proyectar cada uno de estos por separado. A la derecha se tiene un apartado con la lista de caminos para proyección de información de performance donde puede verse los 3 caminos más frecuentes y con el botón "Project" proyectarlo.

En la figura 120 se puede ver la descripción de cada uno de los valores de las actividades desplegadas en el modelo. En una actividad se tiene información de:

- La probabilidad de la tarea en porcentaje
- La frecuencia de aparición de la actividad
- Tiempo total: si es verde es relativamente bajo, si es amarillo relativamente moderado y si es rojo relativamente alto.
- Tiempo de trabajo: si es verde es relativamente bajo, si es amarillo relativamente moderado y si es rojo relativamente alto.
- Tiempo de sincronización: si es verde es relativamente bajo, si es amarillo relativamente moderado y si es rojo relativamente alto.
- Tiempo de espera: si es verde es relativamente bajo, si es amarillo relativamente moderado y si es rojo relativamente alto.

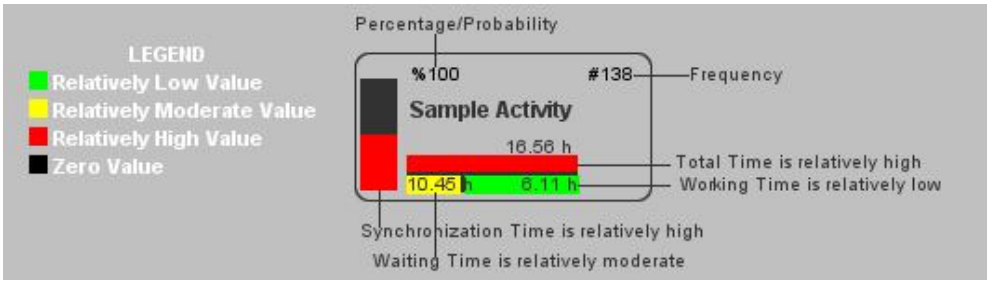


Figura 120 descripción de cada uno de los valores de las actividades *BPMN analysis*

Al costado del display se puede ver indicadores de performance y concordancia como se muestra en la figura 121.

Información de performance:

- Cantidad de casos
- Tasa de arribo por día
- Promedio de ejecución
- Tiempo mínimo de ejecución
- Tiempo máximo de ejecución

Información de concordancia

- Promedio de fitness
- Fitness menor
- Fitness Mayor

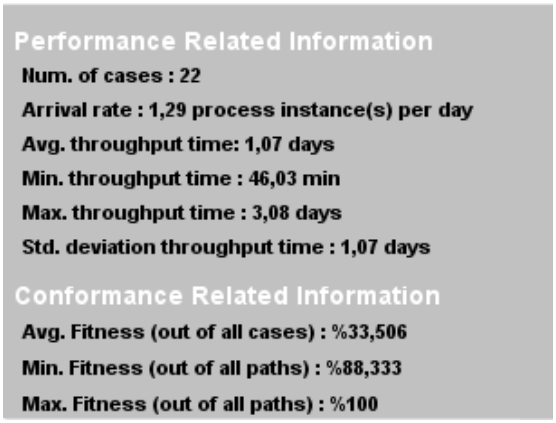


Figura 121 indicador de performance y de concordancia.

A.5.3.1 Solicitud de compra

Si aplicamos la técnica de BPMN analysis sobre el log Solicitud de compra, se visualizará el modelo en notación BPMN y nos permitirá hacer análisis de performance figura 122. En la figura 122 también puede verse que los arcos varían su grosor. Por ejemplo el flujo de notificar rechazo ocurre menos veces que el de solicitar cotizaciones y orden de compra.

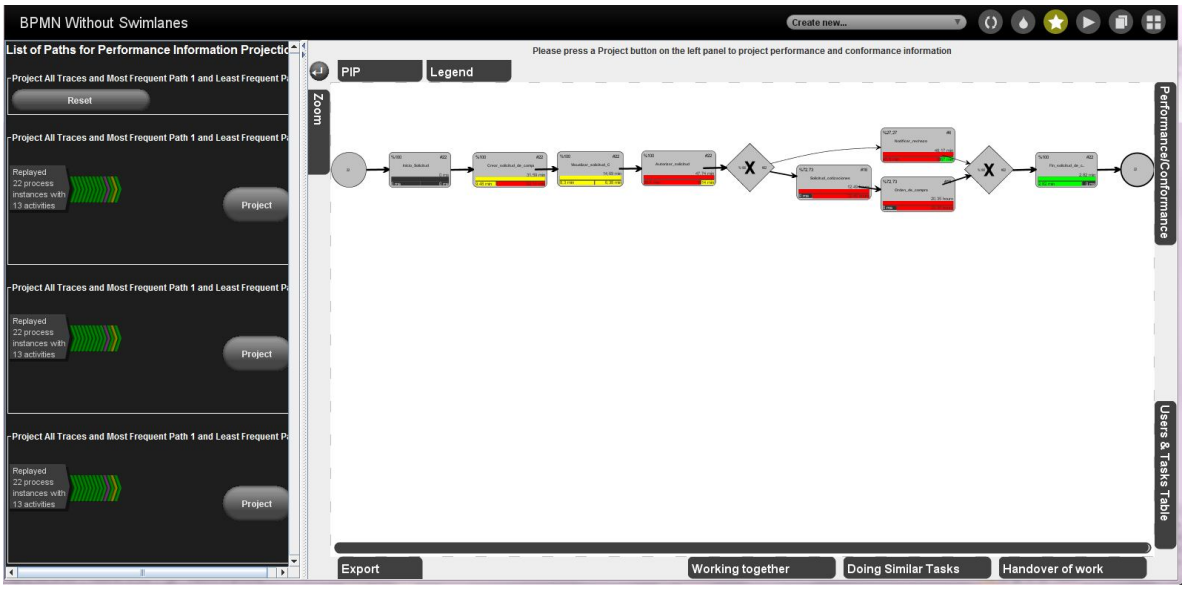


Figura 122 Aplicación de la técnica *BPMN analysis* sobre el proceso de solicitud de compra.

Como puede verse en el modelo las tareas que más tiempo de trabajo tienen son la de solicitud de cotizaciones y orden de compra que son los dos subprocesos. Estas dos actividades no tienen tiempo de espera porque son subprocesos que se inician automáticamente.

La tarea con menos tiempo de trabajo es la de notificar rechazo pues en ella no debe hacerse más que leer la leyenda que informa el rechazo.

La tarea con mayor tiempo de espera es la de autorizar solicitud y es aquí donde se puede producir un cuello de botella ya que se requiere de la interacción del participante jefe.

Mediante las redes de trabajo se puede visualizar el pasaje de trabajo entre los participantes, como en la figura 123 se puede ver perfectamente que mgerez solo toma contacto con lramiez, al igual que abora, mientras que lramirez tiene contacto con los dos empleados administrativos.

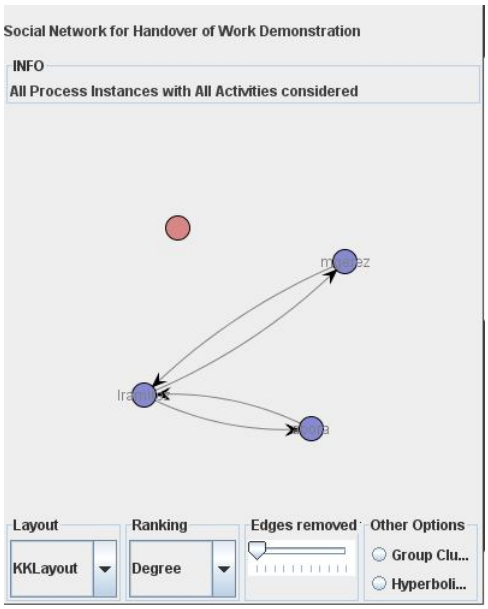


Figura 123 red de trabajo social de pasaje de trabajo entre los participantes

En la figura 124 se puede ver la red de trabajo social de los participantes que realizan tareas similares. Siendo abora y mgerez los que realizan siempre las mismas tareas pudiéndose así identificar dos roles entre los participantes, el rol de empleado operario y el de jefe.

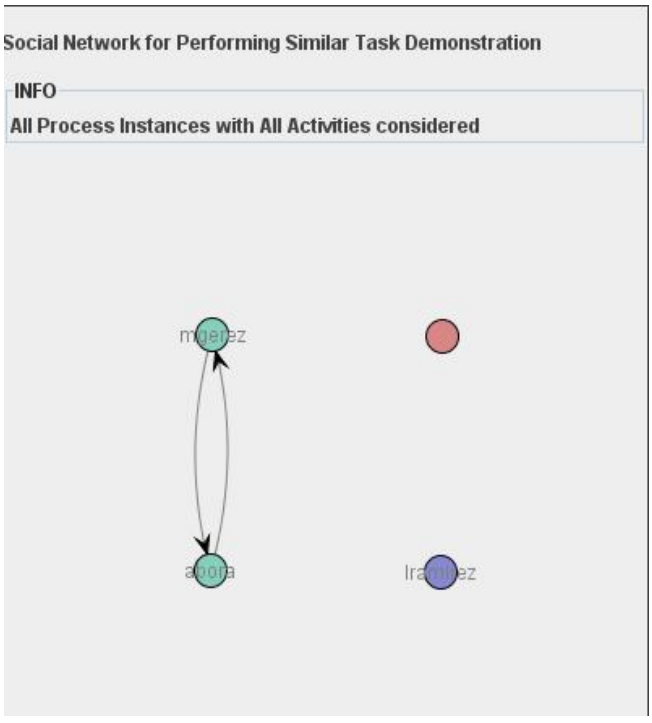


Figura 124 red de trabajo social de los participantes que realizan tareas similares.

A.5.3.2 Solicitud de cotizaciones

Si aplicamos la técnica de BPMN analysis sobre el log Solicitud de cotizaciones, se visualizará el modelo en notación BPMN y nos permitirá hacer análisis de performance

figura 125. En la figura 125 también puede verse que los arcos varían su grosor. Por ejemplo el flujo de recibir cotizaciones y completar cotización se puede ver más grueso que el loop de vuelta por que son menos las veces en las que se cotizan muchos proveedores.

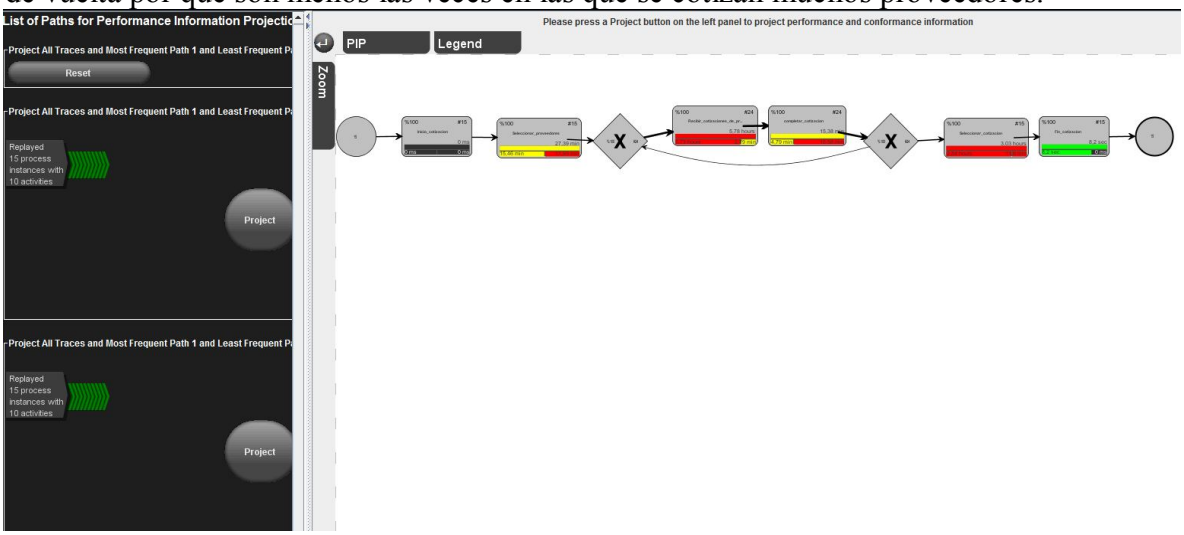


Figura 125 Aplicación de la técnica *BPMN analysis* sobre el proceso de solicitud de cotizaciones.

Como puede verse en el modelo la tarea que más tiempo de trabajo tiene es la del proceso de selección de proveedores.

La tarea con menos tiempo de trabajo es la de seleccionar cotización.

La tarea con mayor tiempo de espera es la de recibir cotizaciones de proveedor porque se debe esperar a que los proveedores se pongan en contacto para enviar la cotización de los productos.

Mediante las redes de trabajo se puede visualizar el pasaje de trabajo entre los participantes, como en la figura 126 se puede ver perfectamente que alara y fbravo son los únicos participantes del proceso pero no tienen tareas específicamente asignadas, es decir no hay pasaje de trabajo entre los participantes.

En la figura 127 se puede ver la red de trabajo social de los participantes que realizan tareas similares. Siendo que alara y fbravo los que realizan siempre las mismas tareas pudiéndose así identificar un role entre los participantes, el rol de empleado administrativo.

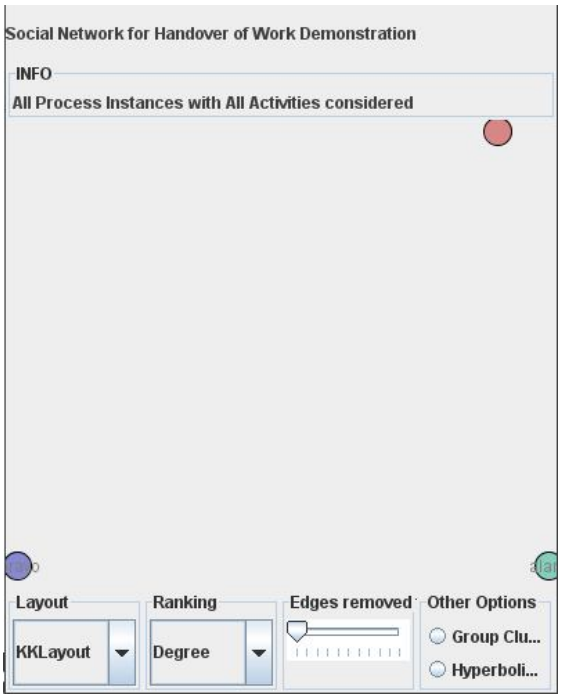


Figura 126 red de trabajo social de pasaje de trabajo entre los participantes

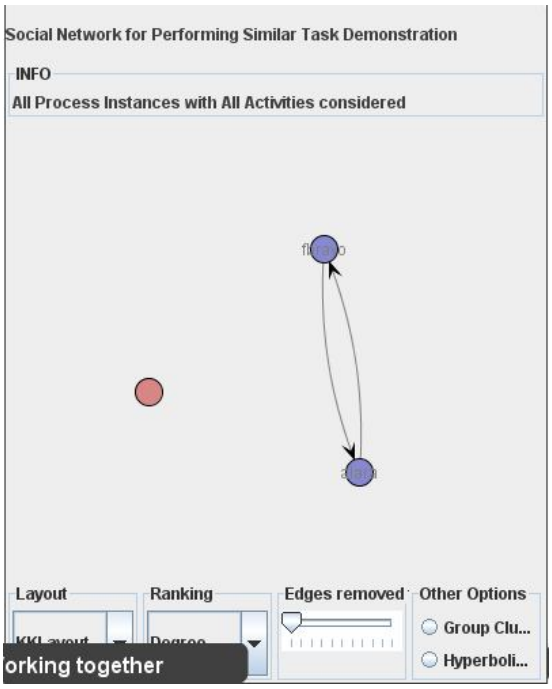


Figura 127 red de trabajo social de los participantes que realizan tareas similares.

A.5.3.3 Orden de compra

Si aplicamos la técnica de BPMN analysis sobre el log Orden de compra, se visualizará el modelo en notación BPMN y nos permitirá hacer análisis de performance

figura 128. En la figura 128 también puede verse que los arcos varían su grosor. Se puede observar que el camino de orden de compra que no requiere aprobación es mucho más frecuente que el que requiere una aprobación.

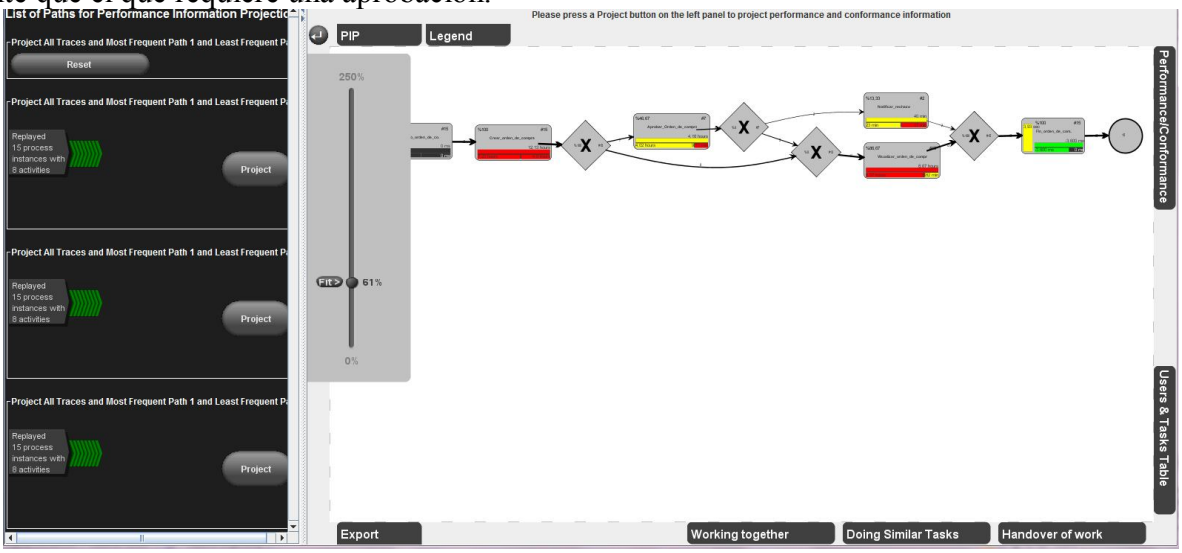


Figura 128 Aplicación de la técnica *BPMN analysis* sobre el proceso de solicitud de cotizaciones.

Como puede verse en el modelo la tarea que más tiempo de trabajo tiene es la crear orden de compra.

La tarea con menos tiempo de trabajo es la de verificar orden de compra ya que lo único que debe hacerse es verificar que los datos de la orden de compra sean correctos.

La tarea con mayor tiempo de espera también es la de verificar orden de compra ya que en los casos en los que la orden de compra debe aprobarse el empleado administrativo puede demorar en recibir la respuesta y realizar el chequeo

La red de trabajo social que indica el flujo de trabajo entre los participantes se observa en la figura 129 y se puede ver perfectamente que alara y fbravo solo intercambian flujo de trabajo con plansky que es el jefe administrativo

En la figura 130 se puede ver la red de trabajo social de los participantes que realizan tareas similares. Siendo que alara y fbravo los que realizan siempre las mismas tareas pudiéndose así identificar dos roles entre los participantes, el rol de empleado administrativo y el de jefe administrativo por plansky.

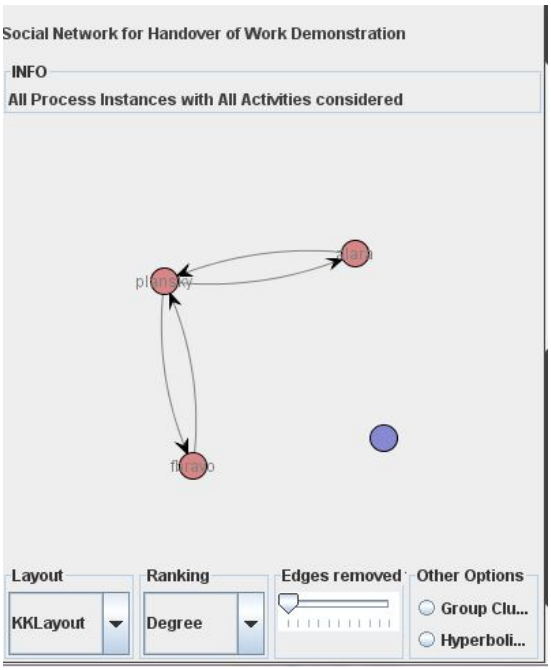


Figura 129 red de trabajo social de pasaje de trabajo entre los participantes

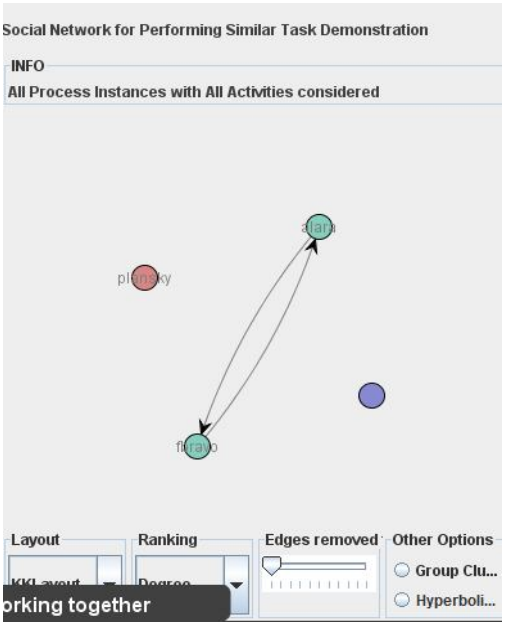


Figura 130 red de trabajo social de los participantes que realizan tareas similares.

A.5.4 Sistema de transición

Mediante el sistema de transición se puede obtener diagramas de nodos y arcos donde los nodos pueden representar los recursos, las actividades, o una conjunción entre los

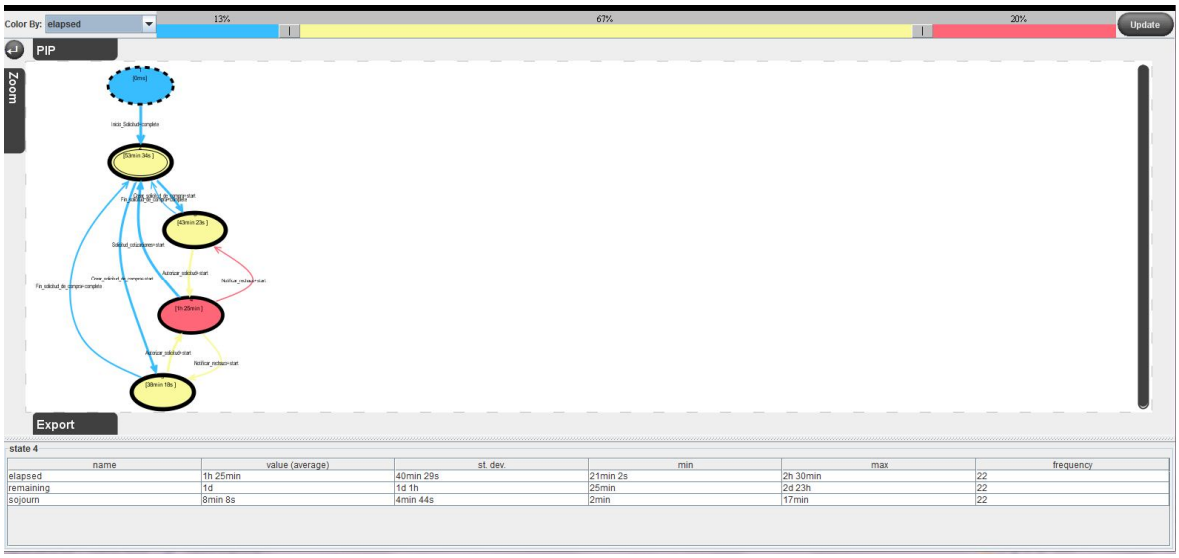
recursos y las actividades. Mientras que los arcos representan a los estados de las actividades.

Los colores de los arcos y de los nodos indican cuales fueron las tareas o recursos que más tiempo intervinieron en el proceso, azul menos tiempo, amarillo un tiempo mayor mientras que rojo son las tareas críticas que más demoraron.

Por el grosor de los arcos se pueden determinar aquellas tareas o recursos más frecuentes.

A.5.4.1 Solicitud de compra

En la figura 131 se puede apreciar un diagrama de transición donde los nodos representan a los recursos y los arcos a los eventos que los relacionan.



name	value (average)	st. dev.	min	max	frequency
elapsed	1h 25min	40min 29s	21min 2s	2h 30min	22
remaining	1d	1d 1h	25min	2d 23h	22
sojourn	8min 8s	4min 44s	2min	17min	22

Figura 131 Diagrama de transición de estados del proceso de solicitud de compra.

Como puede verse el proceso de solicitud de compra tiene cuatro recursos, los 3 participantes humanos, y se identifica un cuarto "participante" que es el automático, porque no tiene nombre. El color de los nodos indica cual es el recurso que más tiempo tardo en completar la tarea, en este caso se puede ver que lramirez (el jefe) es el recurso que más tiempo tardo en completar la tarea. El color y el grosor de los arcos indican también de las eventos cuales fueron las más frecuentes y cuál fue el evento que más tiempo demoro en completarse promedio, en nuestro ejemplo se ve que el inicio de la tarea notificar rechazo fue el que más tiempo tardo en llevarse a cabo.

En la figura 132 se puede ver un diagrama de transición de estados donde los nodos representan a la dupla recurso, actividad mientras que los arcos representan a los eventos que los relacionan. Como puede verse hay un camino para cada recurso, al principio hay dos caminos uno para abora y mgerez, que luego desembocan en lramirez luego se desprenden tres caminos, uno para el rechazo realizado por abora, otro para el rechazo de mgerez y finalmente el de aceptación donde se ejecutan los dos subprocesos de solicitud de cotizaciones y de orden de compra, este camino no tiene recurso puesto que para nuestro proceso son tareas automáticas. Como puede verse mgerez tarda más que abora en completar la tarea de notificar rechazo. Mientras que la tarea que más tarda en completarse

es la de orden de compra y solicitud de cotizaciones (los dos subprocesos) esto puede verse por el color rojo en los arcos.

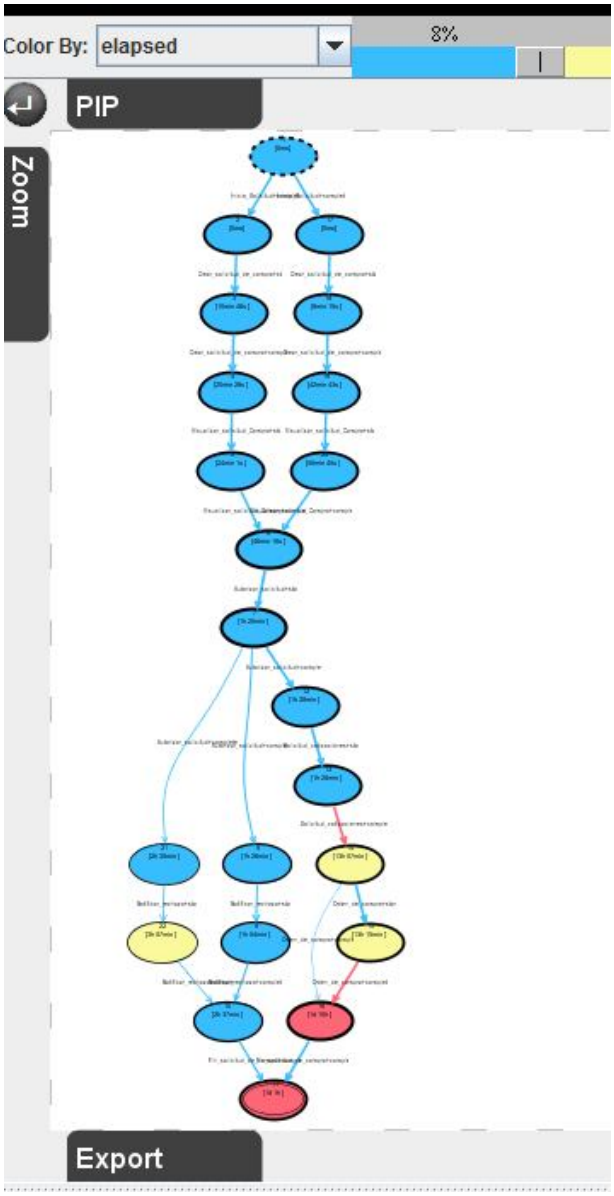


Figura 132 Diagrama de transición de estados del proceso de solicitud de compra

A.5.4.2 Solicitud cotizaciones

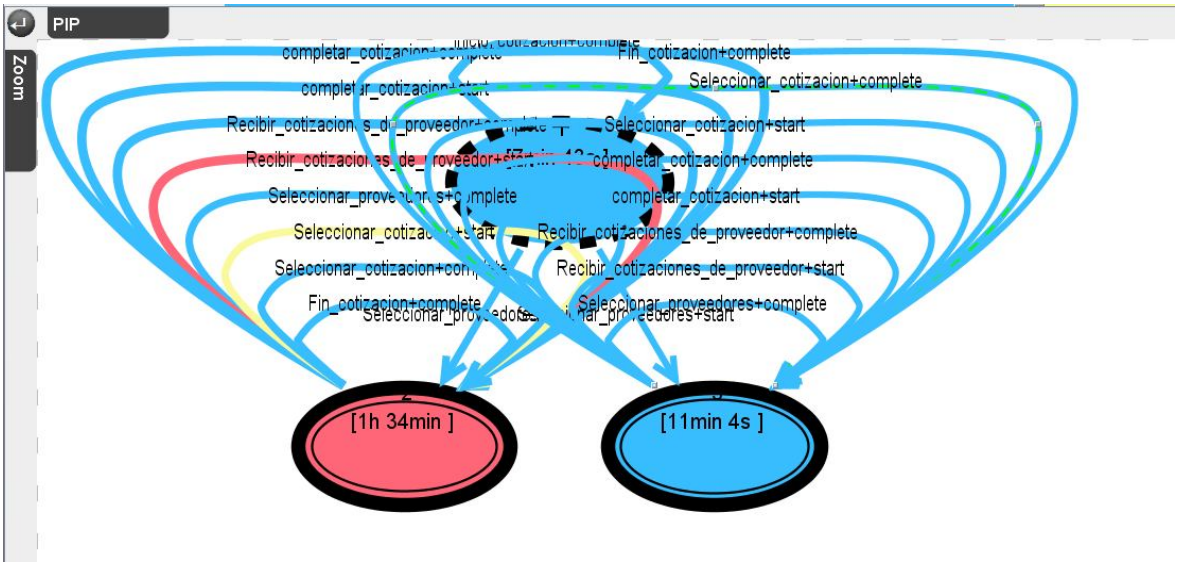


Figura 133 Diagrama de transición de estados del proceso de solicitud de cotizaciones.

En la figura 133 se puede apreciar un diagrama de transición donde los nodos representan a los recursos y los arcos a los eventos que los relacionan.

Como puede verse el proceso de solicitud de cotizaciones tiene tres recursos, los 2 participantes humanos, y se identifica un cuarto "participante" que es el automático, porque no tiene nombre. El color de los nodos indica cual es el recurso que más tiempo tardo en completar la tarea, en este caso se puede ver que fbravo es el recurso que más tiempo tardo en completar la tarea. El color y el grosor de los arcos indican también de las eventos cuales fueron las más frecuentes y cuál fue el evento que más tiempo demoro en completarse promedio, en nuestro ejemplo se ve que el inicio de la tarea recibir cotizaciones de proveedor fue el que más tiempo tardo en llevarse a cabo ya que este es un tiempo externo que no depende del proceso.

En la figura 134 se puede ver un diagrama de transición de estados donde los nodos representan a la dupla recurso, actividad mientras que los arcos representan a los eventos que los relacionan. Como puede verse hay un camino para cada recurso. Se observa que el fbravo fue el único recurso en realizar el loop es decir el único en recibir cotizaciones de más de un proveedor. Se ve también como la tarea que más lo demoro fue la de recibir las cotizaciones de proveedor porque es la que más tiempo de espera tiene.

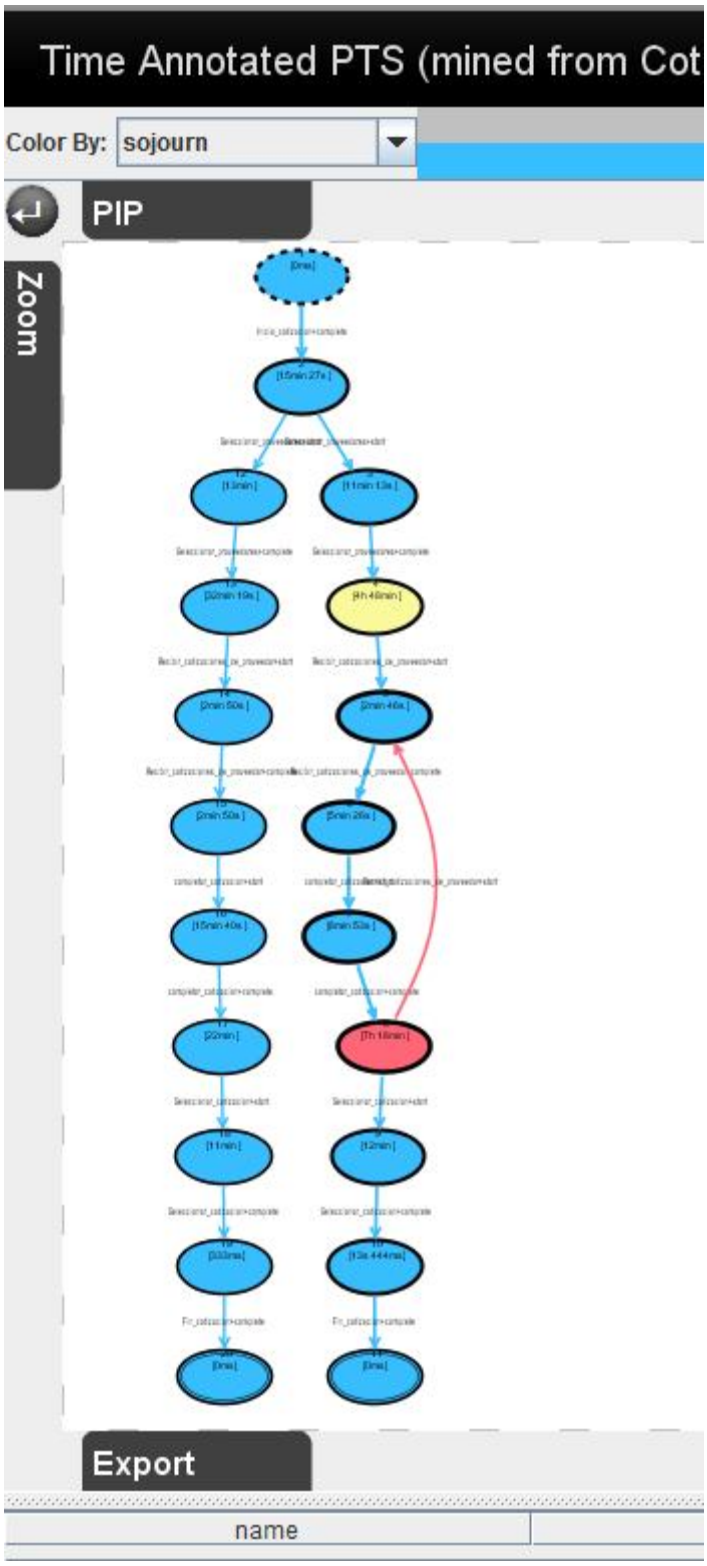


Figura 134 Diagrama de transición de estados del proceso de solicitud de cotizaciones

A.5.4.3 Orden de compra

En la figura 135 se puede apreciar un diagrama de transición donde los nodos representan a los recursos y los arcos a los eventos que los relacionan.

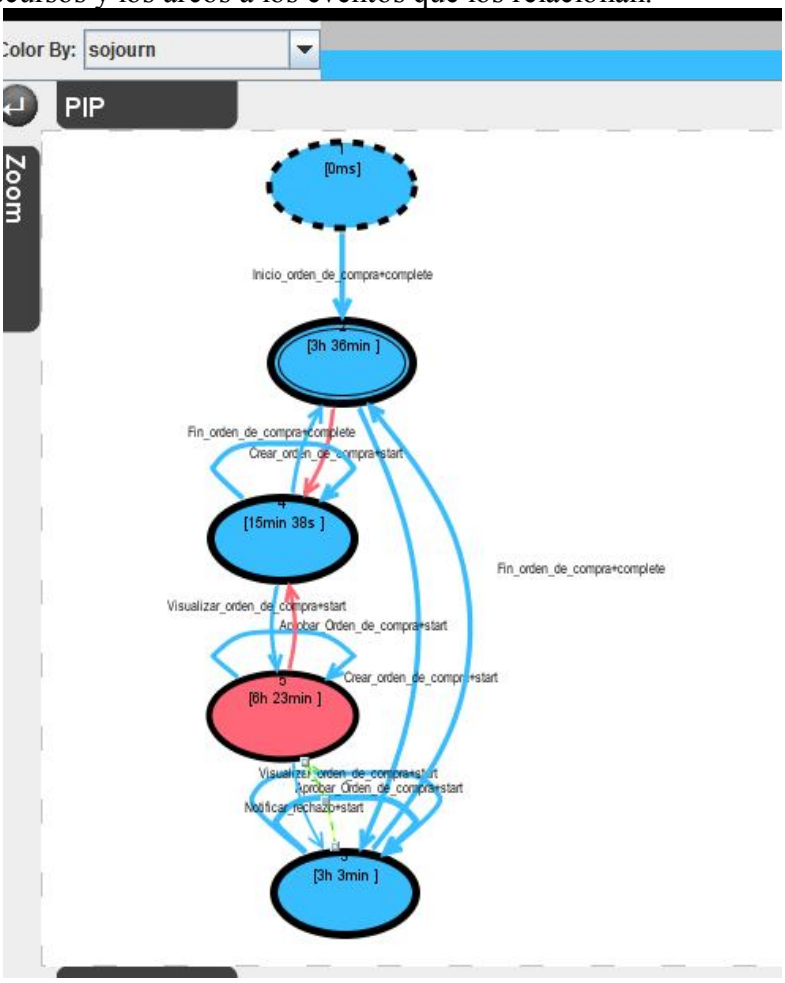


Figura 135 Diagrama de transición de estados del proceso de Orden de compra

Como puede verse el proceso de orden de compra tiene cuatro recursos, los 3 participantes humanos, y se identifica un cuarto "participante" que es el automático, porque no tiene nombre. El color de los nodos indica cual es el recurso que más tiempo tardo en completar la tarea, en este caso se puede ver que plansky (jefe administrativo) es el recurso que más tiempo tardo en completar la tarea. El color y el grosor de los arcos indican también de las eventos cuales fueron las más frecuentes y cuál fue el evento que más tiempo demoro en completarse promedio, en nuestro ejemplo se ve que el inicio de las tareas aprobar orden de compra y crear orden de compra son las que más tiempo tardaron en llevarse a cabo ya que dependen de la interacción del jefe administrativo.

En la figura 136 se puede ver un diagrama de transición de estados donde los nodos representan a la dupla recurso, actividad mientras que los arcos representan a los eventos que los relacionan. Como puede verse hay un camino para cada recurso. Se ve aquí también como la tarea que más lo demoro fue aprobar orden de compra.

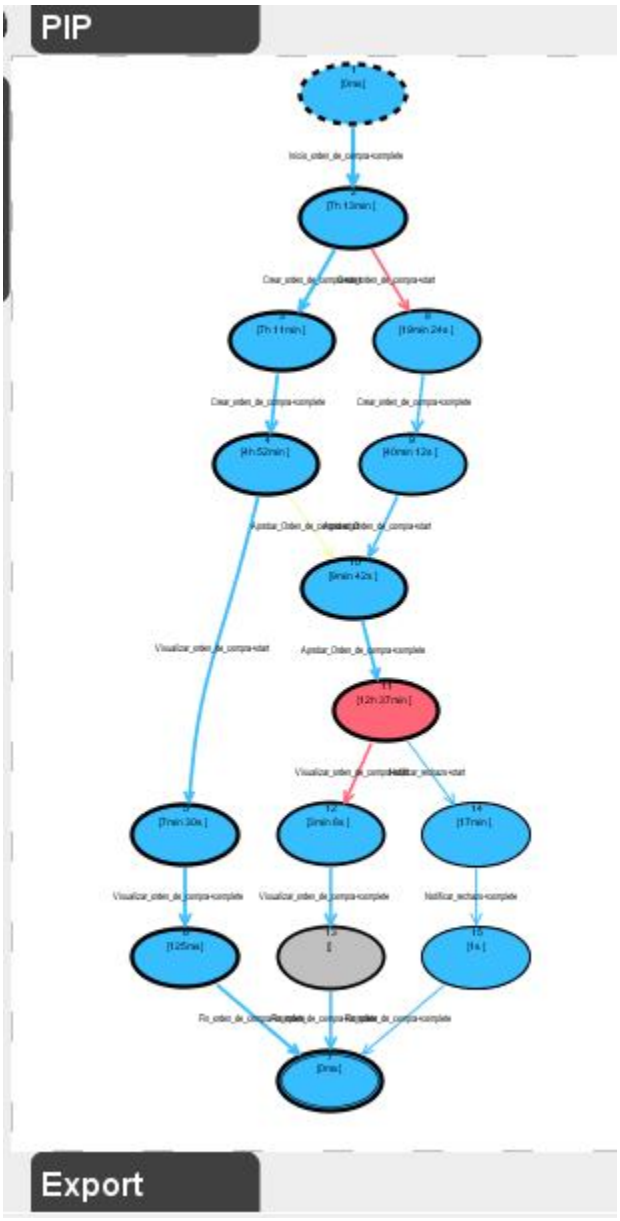


Figura 136 Diagrama de transición de estados del proceso de orden de compra

A.5.5 Fuzzy miner

Eta técnica permite producir un modelo fuzzy permitiendo visualizar las actividades más importantes de acuerdo a un grado de cauterización, también permite visualizar los flujos entre las actividades a menos y mayor detalle. El grosor de los arcos nos indica la frecuencia de ocurrencia que tiene ese camino. La técnica de *Fuzzy miner* también permite animar sobre una instancia de modelo fuzzy la ejecución del log de eventos permitiendo ver a simple vista la frecuencia de ejecución de las actividades y detectar posibles puntos de demora.

A.5.5.1 Solicitud de compra

En la figura 137 se observa un modelo por defecto del proceso de solicitud de compra.

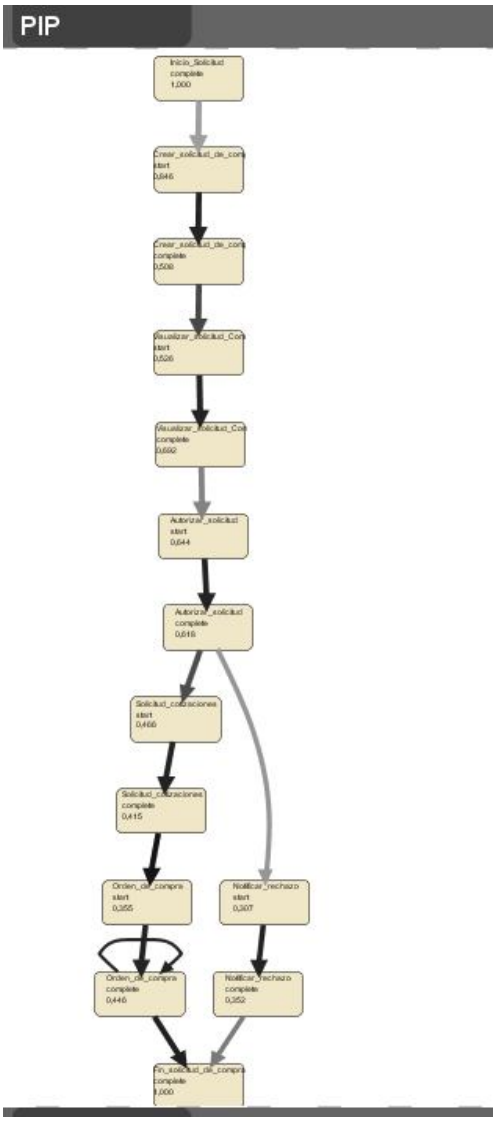


Figura 137 Modelo fuzzy aplicado al proceso de solicitud de compra

Si quisiéramos clusterizar las actividades según la frecuencia de utilización obtendríamos un modelo como el de la figura 138. Mientras que si quisiéramos ver con más detalle el flujo entre las actividades obtendríamos un modelo como el de la figura 139.

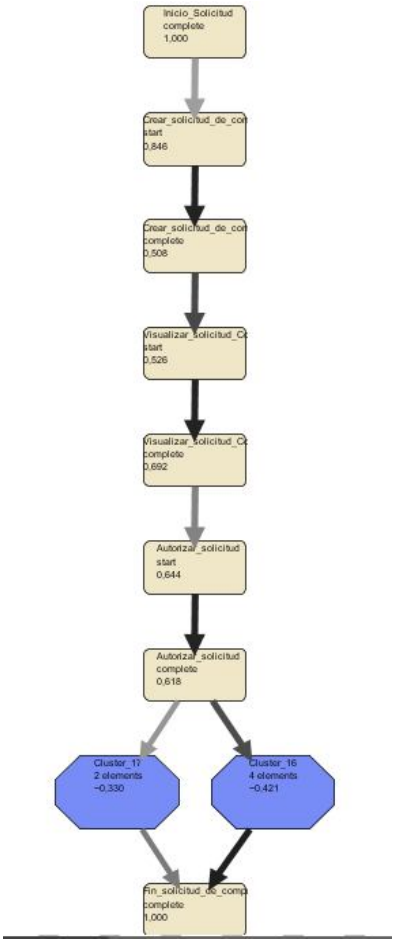


Figura 138 Modelo fuzzy aplicado al proceso de solicitud de compra clisterizando las actividades.

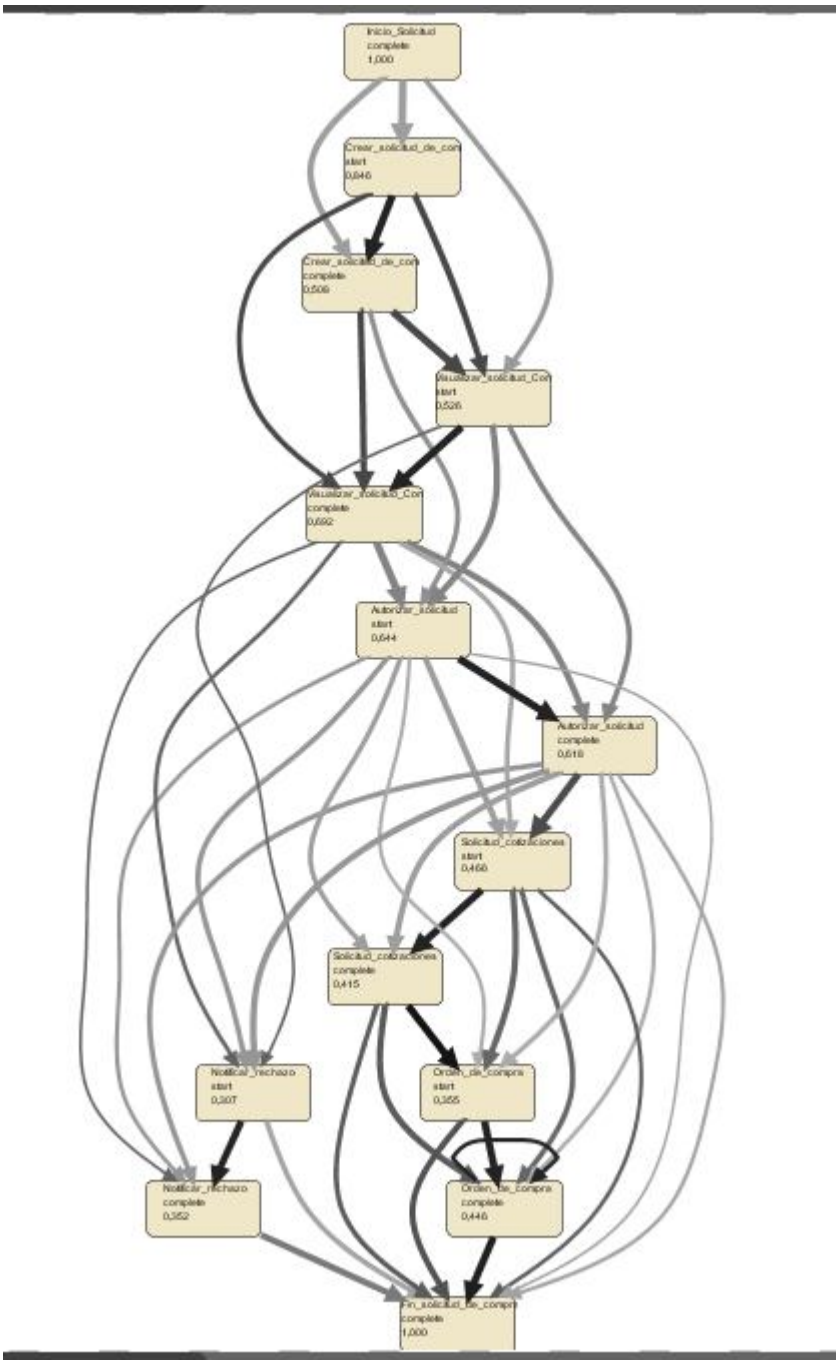


Figura 139 modelo fuzzy aplicado al proceso de solicitud de compra con más detalle de los caminos tomados.

Mediante la animación del modelo fuzzy se puede ver en nuestro proceso de solicitud de compra como el punto de demora está en la autorización de la solicitud y en los subprocesos de cotizaciones y orden de compra. En la figura 140 se muestra un ejemplo de la animación mientras corren los casos de la instancia de proceso.

Figura 140 Animación de un modelo fuzzy del proceso de solicitud de compra.

A.5.5.2 Solicitud de cotizaciones

En la figura 141 se observa un modelo por defecto del proceso de solicitud de cotizaciones.

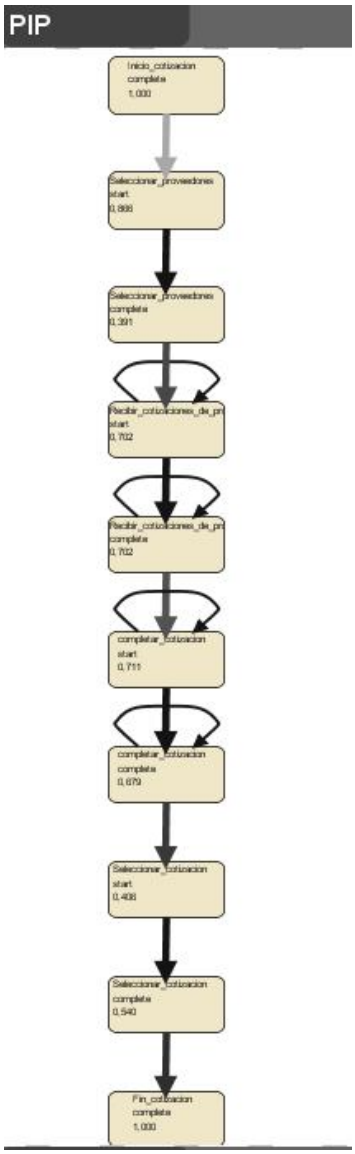


Figura 141 Modelo fuzzy aplicado al proceso de solicitud de cotizaciones

Si quisiéramos clusterizar las actividades según la frecuencia de utilización obtendríamos un modelo como el de la figura 142. Mientras que si quisiéramos ver con más detalle el flujo entre las actividades obtendríamos un modelo como el de la figura 143.

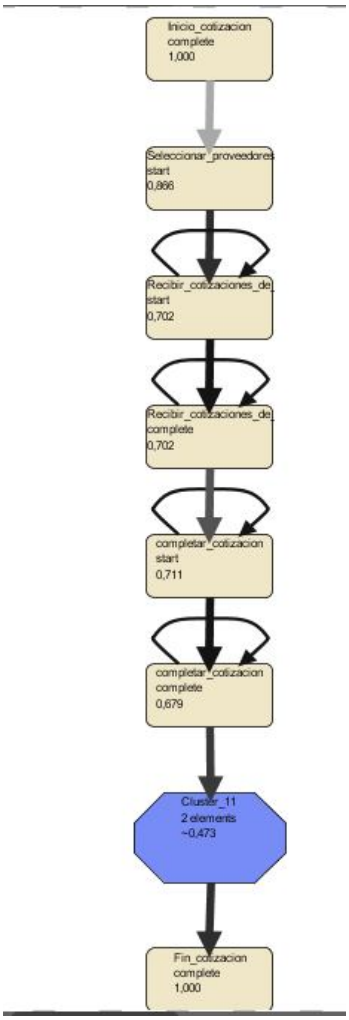


Figura 142 Modelo fuzzy aplicado al proceso de solicitud de cotizaciones clisterizando las actividades.

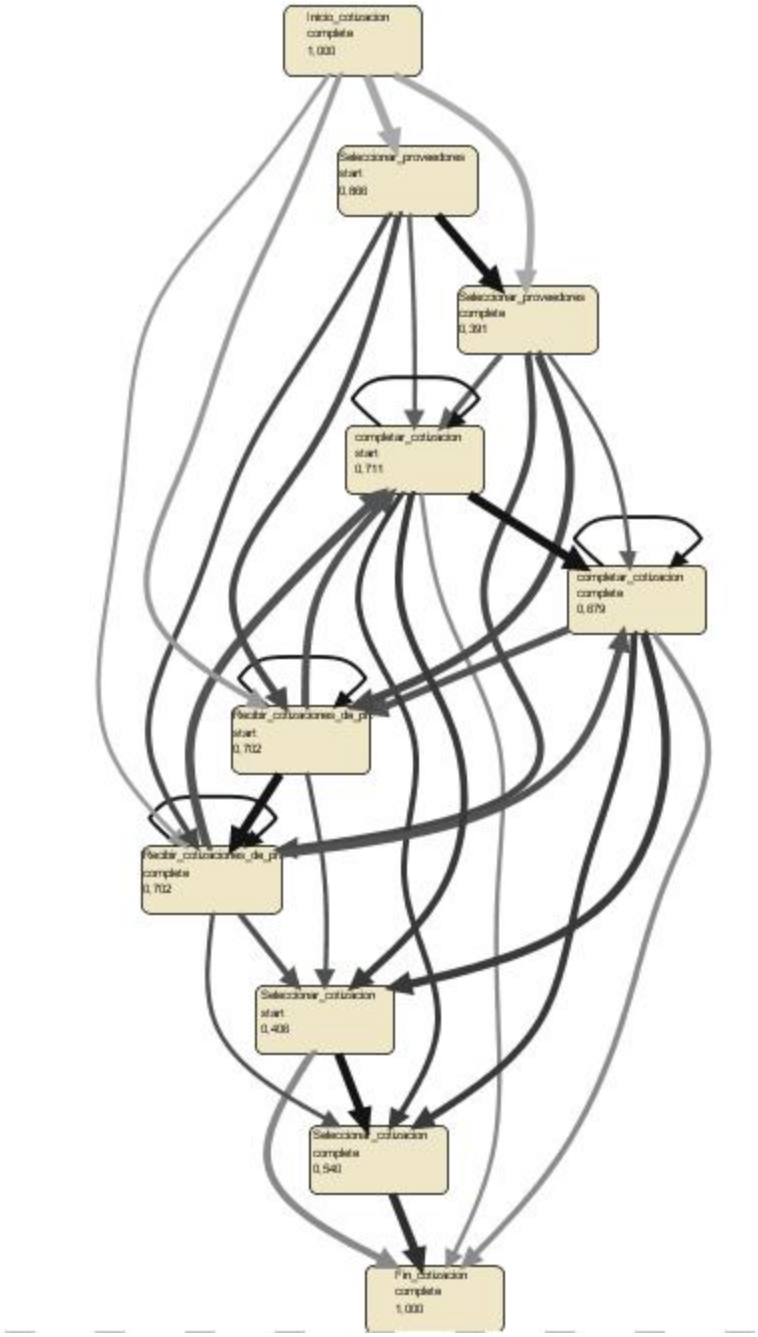


Figura 143 modelo fuzzy aplicado al proceso de solicitud de cotizaciones con más detalle de los caminos tomados.

Mediante la animación del modelo fuzzy se puede ver en nuestro proceso de solicitud de cotizaciones como el punto de demora está en la tarea de recibir cotizaciones de proveedor, ya que se debe esperar a que el proveedor se ponga en contacto para pasar la cotización. En la figura 144 se muestra un ejemplo de la animación mientras corren los casos de la instancia de proceso.

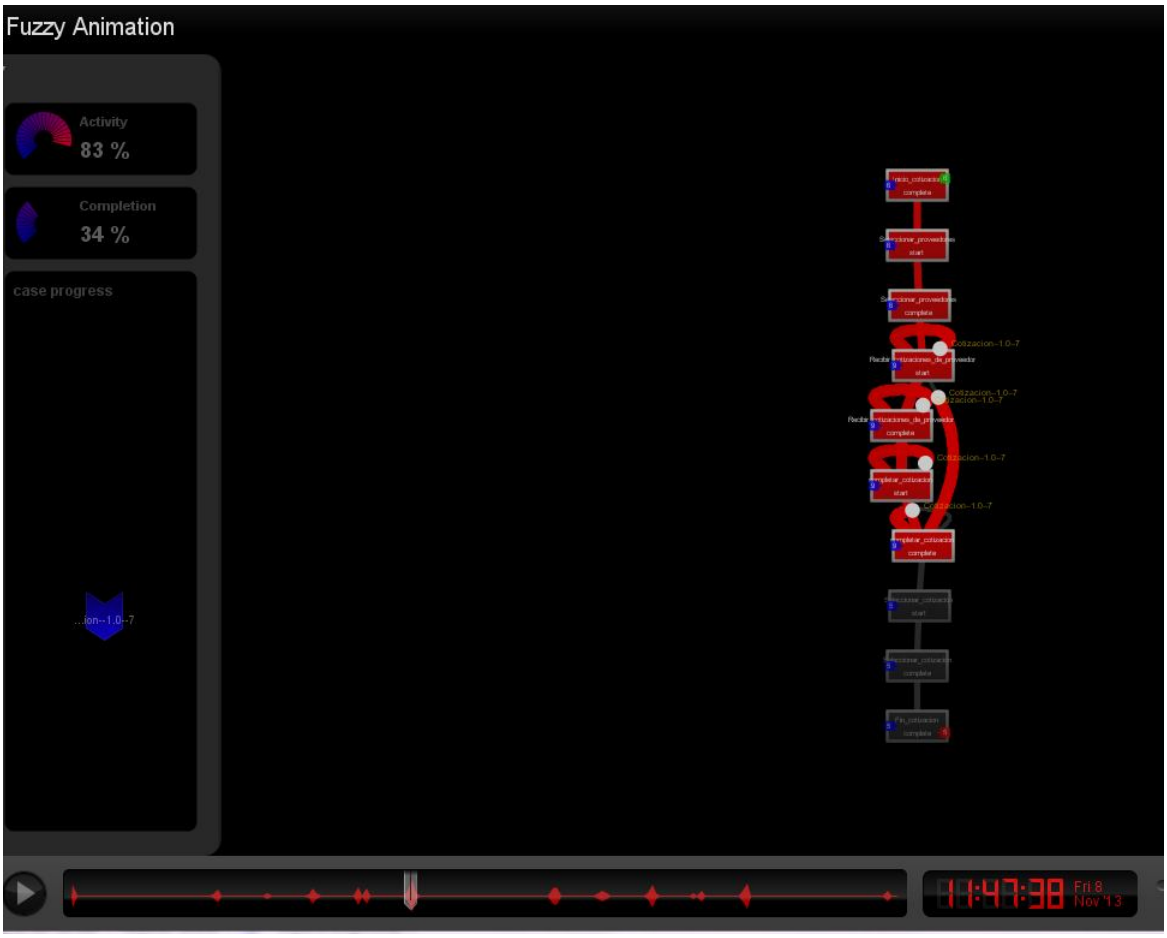


Figura 144 Animación de un modelo fuzzy del proceso de solicitud de cotizaciones

A.5.5.3 Orden de compra

En la figura 145 se observa un modelo por defecto del proceso de orden de compra.

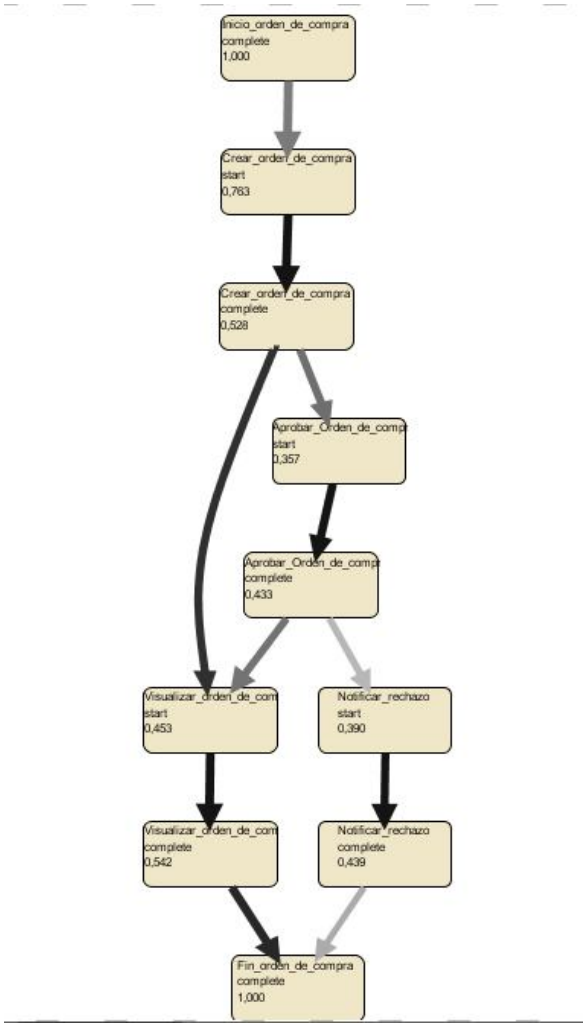


Figura 145 Modelo fuzzy aplicado al proceso de orden de compra

Si quisiéramos clusterizar las actividades según la frecuencia de utilización obtendríamos un modelo como el de la figura 146. Mientras que si quisiéramos ver con más detalle el flujo entre las actividades obtendríamos un modelo como el de la figura 147.

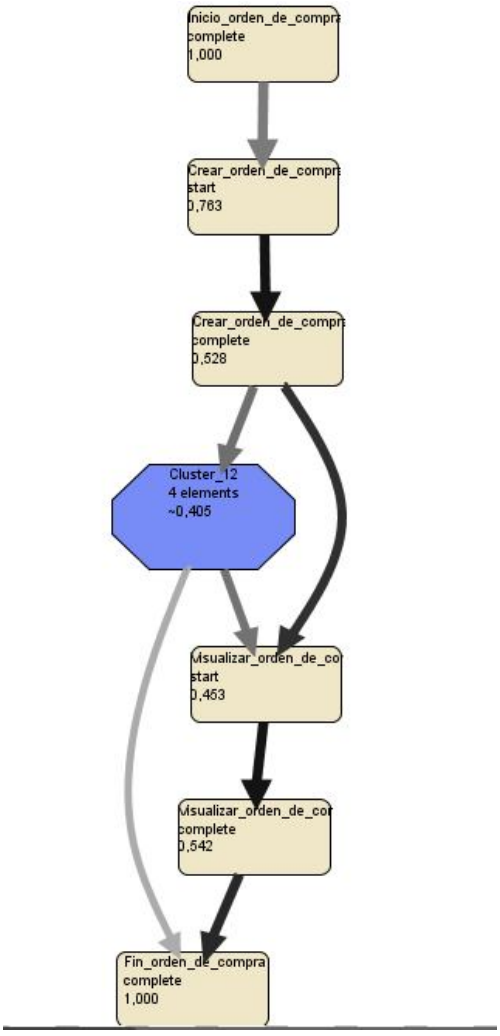


Figura 146 Modelo fuzzy aplicado al proceso de orden de compra clisterizando las actividades.

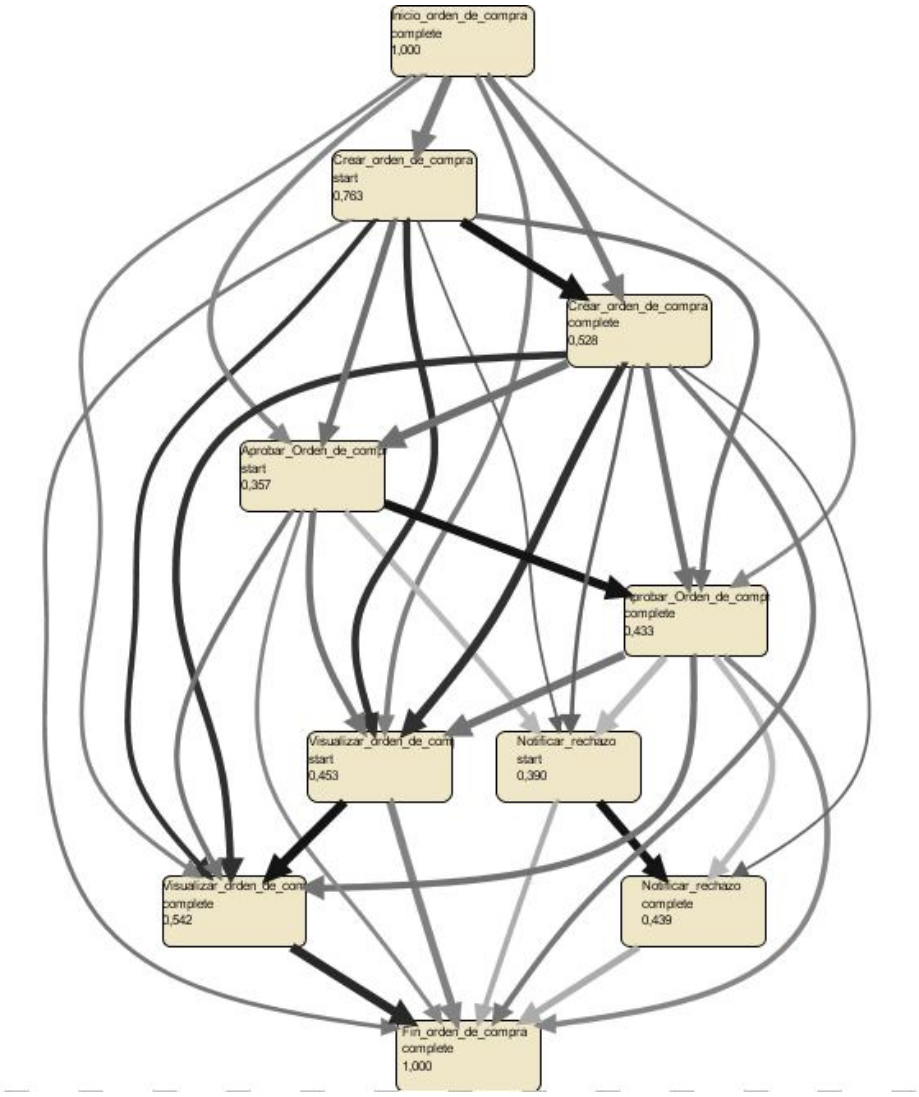


Figura 147 modelo fuzzy aplicado al proceso de orden de compra con más detalle de los caminos tomados.

Mediante la animación del modelo fuzzy se puede ver en nuestro proceso de orden de compra como el punto de demora está en la autorización de la orden de compra. En la figura 148 se muestra un ejemplo de la animación mientras corren los casos de la instancia de proceso.

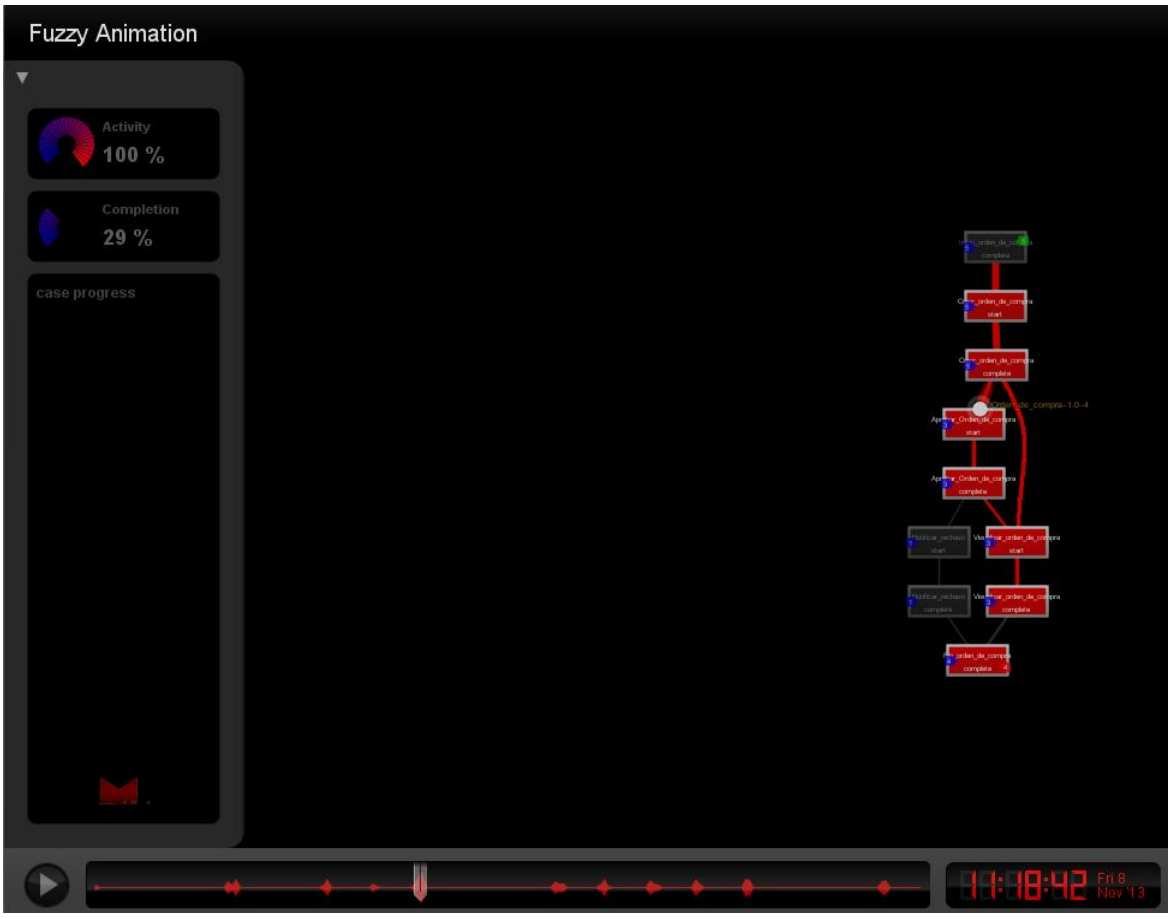


Figura 148 Animación de un modelo fuzzy del proceso de orden de compra

Referencias

[1] Mathias Waske – "*BUSINESS PROCESS MANAGEMENT* - Concepts, Languages, architectures". Springer-Verlag Berlin Heidelberg 2007

[2] "Process BPM Basics For Dummies®, Software AG Special Edition". Kiran Garimella Michael Lees Bruce Williams: Published by Wiley Publishing, Inc., Indianapolis, Indiana. 2008

[3] "*Process Mining* Conformance and Extension". O. by Anne Rozinat. - Eindhoven: Technische Universiteit Eindhoven, 2010. - Proefschrift.

[4] "*Process Mining* Discovery, Conformance and Enhancement of Business Processes". Wil M.P. van der Aalst Department Mathematics & Computer Science Eindhoven University of Technology Den Dolech 2 5612 AZ Eindhoven The Netherlands. 2011.

[5] "Using *Process Mining* to Bridge the Gap between BI and BPM". Wil M.P. van der Aalst Department Mathematics & Computer Science Eindhoven University of Technology Den Dolech 2 5612 AZ Eindhoven The Netherlands. 2011.

[6] *Process Mining* org site: http://www.processmining.org/

[7] OMG *Business Process Management* Initiative. www.bpmn.org.

[8] Sitio de Bonita Soft http://es.bonitasoft.com/

[9] "XES, XESame, and ProM 6". H.M.W. Verbeek, Joos C.A.M. Buijs, Boudewijn F. van Dongen, and Wil M.P. van der Aalst. Springer-Verlag Berlin Heidelberg 2010

[10] "Understanding Spaghetti Models with Sequence Clustering for ProM". Gabriel M. Veiga and Diogo R. Ferreira

[11] "Genetic-based Anomaly Detection in Logs of Process Aware Systems". Hanieh Jalali, Ahmad Baraani. World Academy of Science, Engineering and Technology 40 2010.

[12] "Discovering Hierarchical Process Models Using ProM". R.P. Jagadeesh Chandra Bose, Eric H.M.W. Verbeek and Wil M.P. van der Aalst.

[13] "*Process Mining* Applied to the BPI Challenge 2012: Divide and Conquer While Discerning Resources". R.P. Jagadeesh Chandra Bose and Wil M.P. van der Aalst.

[14] "A Comprehensive Benchmarking Framework (CoBeFra) for Conformance Analysis between Procedural Process Models and Event Logs in ProM". Seppe K.L.M. vanden Broucke, Jochen De Weerdty, Jan Vanthienen, and Bart Baesens. 2013.

[15] “Merging Computer Log Files for *Process Mining*: an Artificial Immune System Technique”. Jan Claes and Geert Poels.

[16] “Análisis metodológico para la utilización de *Process Mining* como tecnología de optimización y respaldo de la implementación de procesos de negocio bajo el marco de BPM”. Virginia María Magliano, Mg. Patricia Bazán, Lic. José Martinez Garro. 2013.

[17] “ *Business Process Management Systems* ” James. F. Chang. Auerbanch Publications. 2006.

[18] “Sistemas y Organizaciones” Ing. Emilio Lorenzon. 2009.