Soporte a la Evolución de Familias de Procesos de Negocio Mediante Patrones de Cambio

Viacheslav Karpov

Trabajo Final de Master

Máster en Ingeniería del Software, Métodos formales y Sistemas de Información
Departamento de Sistemas Informáticos y Computación

Dirigido por:
Vicente Pelechano Ferragud
Victoria Torres Bosch

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Abril 2014

Resumen

Los procesos de negocio existen en diferentes variantes dependiendo de los distintos contextos de aplicación (ej. el proceso de facturación online vs presencial en el aeropuerto). Estas variantes forman, en conjunto, una *familia de procesos.* Existen varias propuestas para gestionar estas familias de procesos mediante extensiones de lenguajes de modelado de procesos. Sin embargo, para mejorar y facilitar la gestión de la variabilidad en familias de procesos de negocio, se han definido un conjunto de *patrones de cambio específicos para la variabilidad en los procesos de negocio*. Estos patrones son genéricos e independientes del lenguaje de modelado de procesos por lo que pueden utilizarse en combinación con cualquier propuesta para gestionar las variantes (por ej. BPMN, EPC, YAWL, Diagramas de actividad de UML, Redes de Petri, etc.). En este trabajo se propone la implementación de estos patrones de cambio mediante la extensión de la Plataforma Experimental Cheetah (*Cheetah Experimental Platform*, CEP)[1]. Concretamente, se va a extender el editor gráfico de CEP añadiendo nueva funcionalidad que permita la utilización de los patrones de cambio definidos. Gracias a esta implementación, el modelado de los procesos variables resulta más cómodo, menos costoso y más fácil.

Palabras clave

Familias de procesos, patrones de cambio, variabilidad de procesos, procesos de negocio

[1] http://bpm.q-e.at/?page_id=56

Índice

Índice de Figuras

Índice de Tablas

1 INTRODUCCION

Actualmente, la gestión de procesos de negocio es crucial para las empresas. El concepto de *proceso de negocio* nace de que cada producto o servicio que una empresa produce o presta es el resultado de un conjunto de actividades realizadas de forma coordinada (ej. facturación de un billete de avión). Los *modelos de procesos de negocio* son la representación gráfica de esa coordinación de actividades. Sin embargo, construir modelos de proceso de negocio supone un gran reto ya que está altamente ligado a la naturaleza del dominio que se está modelando. Por ejemplo, un proceso de fabricación de un armario y un proceso de facturación de un billete es diferente. Además de estas diferencias, dentro de un mismo dominio (ej. el dominio aéreo) también se presentan variaciones de un mismo proceso (ej. facturación online, en el mostrador en el aeropuerto, o en la máquina de auto-servicio).

Como resultado de estas variaciones dentro de un mismo dominio, para un tipo de proceso en concreto (ej. facturación de un billete de avión) existen una multitud de *variantes de proceso,* cada una de los cuales es válida en un contexto en particular. Concretamente, dichas variantes persiguen el mismo (o similar) objetivo de negocio (ej. facturación de un billete), pero al mismo tiempo difieren según el contexto de aplicación en el que se utilizan, como por ejemplo, distinta legislación, servicios, o clientes. A estas colecciones de variantes de proceso relacionadas se las conoce como *familias de procesos*. En las grandes empresas, una familia de procesos puede estar compuesta por docenas o cientos de variantes. Por ejemplo, en [2] se presenta una familia de procesos para el mantenimiento de vehículos compuesta por más de 900 variantes con diferencias específicas por el taller, el país, y el propio vehículo. Por su parte, en [1] se describe una familia de procesos que incluye más de 90 variantes para la planificación y el manejo de los exámenes médicos. Otro ejemplo, se encuentra en los procesos de facturación de los billetes en los aeropuertos. A continuación, se detalla este proceso para poder utilizarlo como ejemplo práctico a lo largo de todo el documento.

Ejemplo 1: Aunque este proceso sea similar independientemente del aeropuerto de salida y de la aerolínea con la que vuela el pasajero, existen numerosas variaciones según factores distintos. Por ejemplo, la variabilidad puede ser causada por el tipo de facturación (ej., online, en el mostrador, o en la máquina de auto-servicio), que, a su vez, determina el tipo de tarjeta de embarque (ej., electrónica o de papel). Otras causas de variabilidad son el destino del vuelo (ej., para viajar a los EE.UU. es necesario aportar información sobre el alojamiento) y el tipo de pasajero (ej., los menores no acompañados y las personas con discapacidad necesitan asistencia de personal adicional). Dependiendo del tipo de equipaje (ej., equipaje con sobrepeso o equipaje de gran volumen) el proceso cambia ligeramente ya que puede ser necesario pagar algún tipo de cargo adicional. Finalmente, variaciones temporales con respecto al momento en que se abre el proceso de facturación también son posibles (ej., posibilidad de realizar la facturación varios días antes de la salida o apenas unas horas antes del vuelo).

En las Figuras 1 y 2 se muestran seis variantes simplificadas de este proceso representadas en notación de *Modelado de Procesos de Negocio* (*Business Process Modeling Notation,* BPMN [24]). Estas variantes fueron diseñadas conjuntamente con expertos en el dominio de facturación de billetes de avión. Las actividades comunes a todas las variantes están resaltadas en color gris. Las variantes 1 y 2 (ver Figura 1) asumen que el pasajero realiza la facturación online a través de una página web. Primero, el pasajero se identifica y automáticamente se le asigna un asiento regular. En cuanto a la variante 1, un pasajero de clase preferente viaja de Europa a los EE.UU., lo que requiere información sobre el alojamiento y rellenar el sistema electrónico de autorización de viaje (el formulario ESTA). Por último, se imprime una tarjeta de embarque electrónica y el pasajero deja el equipaje en el mostrador de la clase preferente. En cuanto a la variante 2, después de imprimir la tarjeta de embarque, es necesario pagar un cargo adicional en las taquillas de la aerolínea debido al sobrepeso del equipaje. A su vez, para la variante 3, la facturación se realiza en la máquina de auto-servicio y el equipaje se deja en el mostrador de entrega rápida. Para estas tres variantes del proceso, la facturación está disponible 23 horas antes de la salida.

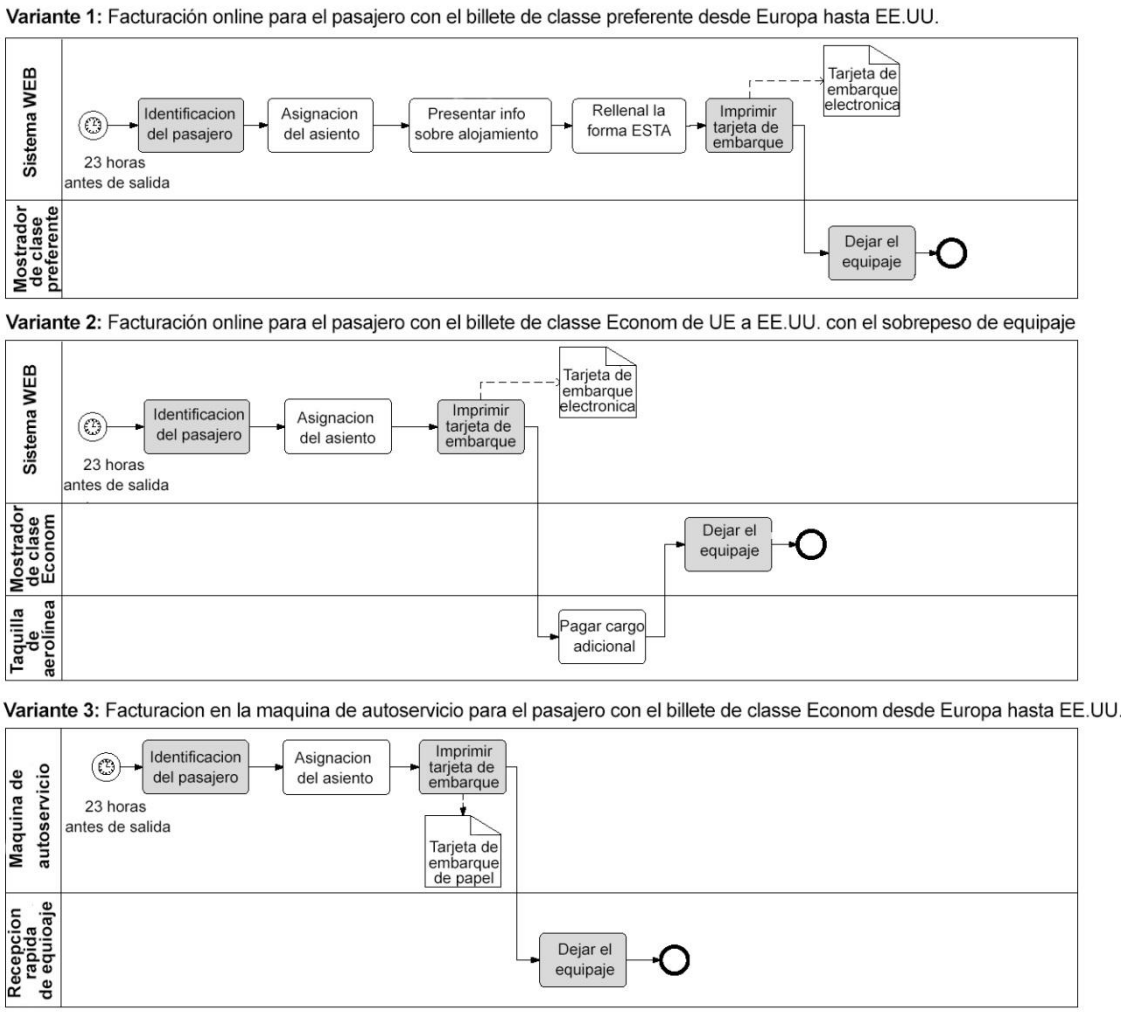


Figura 1: Variantes del proceso de facturación (1)

Por el contrario, las variantes 4-6 (ver Figura 2) representan el proceso realizado directamente en el aeropuerto. Por ejemplo, la variante 4 describe la facturación de un menor que viaja sin acompañante. En esta variante, al menor se le asigna un asiento especial y se rellena un formulario adicional que acompaña su facturación. Además, se necesita una copia de la tarjeta de embarque para el acompañante que vaya con el menor hasta la puerta de embarque. La variante 5 se refiere a un pasajero con discapacidad que necesita la asistencia adicional de una persona que lo acompañe. Finalmente, la variante 6 muestra el proceso de facturación de un pasajero que lleva equipaje grande (de gran volumen). En estas tres variantes del proceso, la facturación solo se puede hacer 3 horas antes de la salida, una vez hayan abierto los mostradores de la aerolínea. Por último, la tarjeta de embarque se imprime en papel.

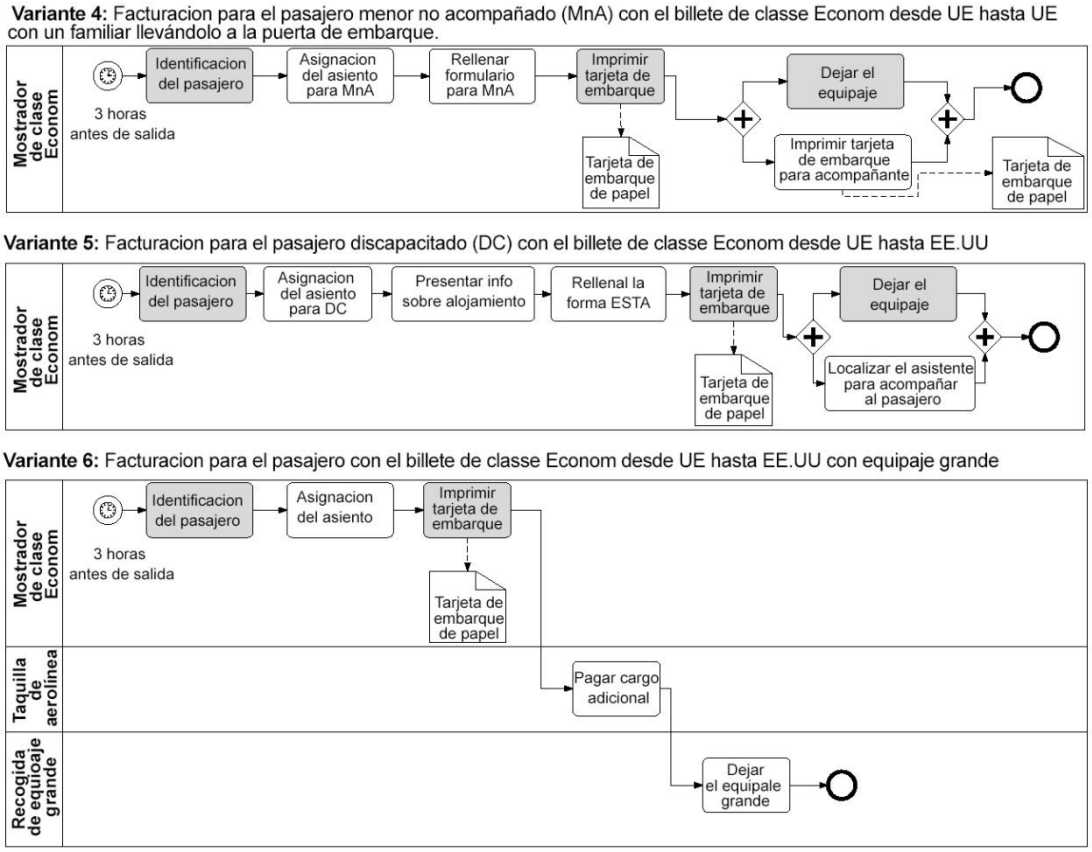


Figura 2: Variantes del proceso de facturación (2)

1.1 Planteamiento del Problema

Diseñar y mantener por separado cada variante del modelo de un proceso como el presentado en el Ejemplo 1 resulta redundante, ineficaz y costoso [2]. Por lo tanto, existe un gran interés en capturar el conocimiento común de los procesos, es decir, las partes comunes de las variantes, sólo una vez y reutilizarlo en términos de *modelos de procesos configurables.* Estos modelos representan la familia de procesos de forma completa y no redundante [2].

Motivado por las limitaciones de los enfoques tradicionales de modelado de los procesos de negocio [2], se han desarrollado diversas propuestas para manejar las familias de procesos [2,3,38,39]. Básicamente, estas propuestas extienden los lenguajes tradicionales de modelado de los procesos de negocio (ej. BPMN) con construcciones

específicas para gestionar la variabilidad (ej. punto de variación). Así, tratando la variabilidad como aspectos de primera clase, estas extensiones ayudan a evitar redundancias, promueven la reutilización y reducen esfuerzos de modelado. Sin embargo, la introducción de construcciones específicas para gestionar la variabilidad implica una complejidad adicional en relación con el lenguaje de modelado utilizado. Para que estas propuestas puedan usarse de manera industrial, la calidad de los modelos de procesos configurables creados es crucial y, por tanto, se necesita un apoyo eficaz a los diseñadores de procesos cuando manejan familias de proceso.

Recientemente, se han introducido con éxito los denominados *patrones de adaptación*, genéricos e independientes del lenguaje, para la creación y la evolución de los modelos (individuales) de procesos de negocio [6,7]. Probados empíricamente en diversos experimentos [5], estos patrones permiten crear y modificar modelos de los procesos de negocio a un nivel alto de abstracción, fomentando la calidad del modelo del proceso resultante, y asegurando su corrección desde su construcción, es decir, el modelo resultante de la aplicación de los patrones de adaptación es correcto. Además, estos patrones de adaptación han servido como base para la implementación de cambios en diferentes etapas del ciclo de vida de los procesos como, por ejemplo, la creación del modelo [6, 13], la configuración del mismo [2], cambios de instancia de proceso en tiempo de ejecución [14, 15], la evolución del modelo [14], la refactorización del modelo [16], la reutilización de los cambios realizados sobre el modelo [17], y la comparación de modelos [18].

Aunque los patrones de adaptación son adecuados para la creación y modificación de los modelos (individuales) de procesos de negocio, no son suficientes para enfrentarse a las necesidades específicas que introducen las familias de proceso [4]. Por ejemplo, estos patrones no cubren las nuevas construcciones del lenguaje que aparecen al gestionar variantes de procesos (ej. puntos de variación). Para solucionar este problema, en [4] se proponen un conjunto complementario de patrones de cambio genéricos, e independientes del lenguaje, específicamente adaptados para las familias de proceso. Usados en combinación con los

patrones de adaptación ya existentes, *los patrones de cambio para las familias de procesos* permitirán la gestión de dichas familias a un alto nivel de abstracción. Por ejemplo, pueden servir como referencia para implementar familias de modelos de procesos, realizar cambios a lo largo de su ciclo de vida, y fomentar la comparación de las propuestas existentes para la variabilidad de los procesos de negocio. Sin embargo, para poder poner en práctica estos patrones de cambio se necesita de herramientas sofisticadas que faciliten y automaticen de manera intuitiva, tanto como se pueda, la aplicación de los patrones de cambio.

1.2 Solución Propuesta

Para facilitar la aplicación de los patrones de cambio para familias de procesos, en este trabajo se propone implementar los mismos sobre el editor gráfico de la *Plataforma Experimental Cheetah* (*Cheetah Experimental Platform*, CEP) (ver Figura 3). Actualmente, CEP es una plataforma que permite modelar y experimentar con modelos (individuales) de procesos de negocio [25]. Básicamente, CEP proporciona componentes para llevar a cabo experimentos controlados a través de la orientación al usuario. Además, CEP proporciona técnicas de evaluación más potentes comparadas con experimentos basados de papel que fomentan el análisis de datos obtenidos del experimento. Por ejemplo, se permite la monitorización de los experimentos paso a paso de tal manera que se pueden obtener datos más pormenorizados y detallados. Así, en CEP se pueden ejecutar experimentos de modelado mitigando los riesgos que ponen en peligro la validez de los datos recogidos.

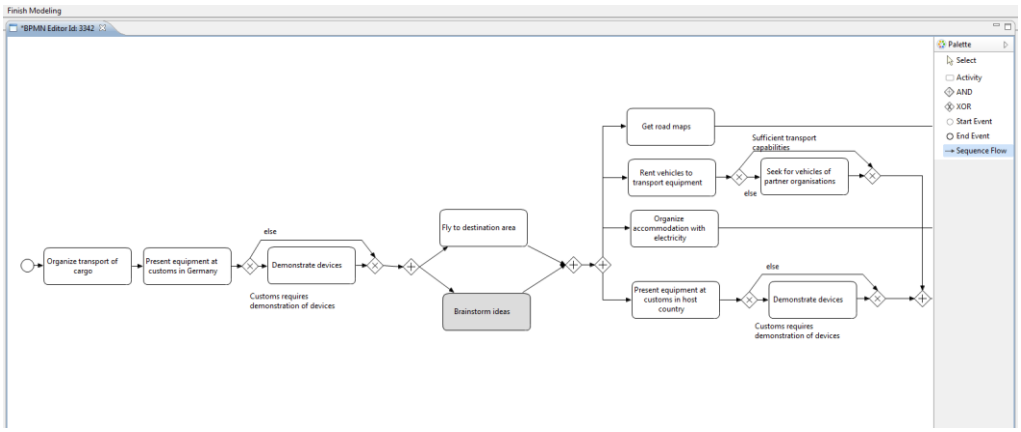


Figura 3: Editor gráfico de CEP

Concretamente, el objetivo de este trabajo es extender el editor gráfico de CEP añadiendo nueva funcionalidad para poder implementar los patrones de cambio de familias de proceso que permitirá el modelado y la evolución de familias de procesos. Gracias a esta implementación, no sólo se facilita y automatiza la aplicación de los patrones de cambio para familias de procesos si no que, además, se permitirá la realización de experimentos controlados que ayuden a determinar la viabilidad de dichos patrones.

1.3 Estructura del Trabajo

El presente documento está organizado de la siguiente manera:

En la Sección 2 se presentan los conceptos básicos de los procesos de negocios necesarios para el entendimiento del trabajo, estos incluyen las primitivas de modelado de procesos de negocio así como el ciclo de vida de dichos procesos. Este último se presentará en el contexto de las familias de procesos de negocio.

En la Sección 3 se aborda el tema de la variabilidad en los procesos de negocio. Para ello en primer lugar se describirán las construcciones del lenguaje identificadas para el manejo de la variabilidad de modelo de procesos. Posteriormente se presentará el conjunto de patrones de cambio para familias de procesos a los que se da soporte en este trabajo.

En la Sección 4 se describe cómo se ha llevado a cabo la implementación de los patrones de cambio en la herramienta CEP. Para ello, en primer lugar se describe el contexto tecnológico en el que se ha llevado a cabo la implementación. Concretamente se presenta el lenguaje de programación en CEP y la plataforma tecnológica que se utiliza para desarrollar la propuesta. Finalmente se describe la implementación realizada. Concretamente, se presenta cómo se han implementado los patrones de cambio, cómo es posible obtener una variante del proceso y la descripción detallada de las herramientas implementadas.

En la Sección 5, se ilustra como a través de un caso de estudio real (el proceso de facturación de un billete de avión) se pueden aplicar los patrones de cambio implementados.

Por último, en la Sección 6 se explican las conclusiones extraídas del trabajo realizado y se detallan las posibles extensiones futuras del mismo.

2 CONTEXTO

En esta Sección se describen los conceptos básicos de los procesos de negocios necesarios para el entendimiento del trabajo. Concretamente, la Sección 2.1 introduce la definición de procesos de negocio mientras que la Sección 2.2 describe brevemente el ciclo de vida de los procesos de negocio focalizándose en la variabilidad de los procesos de negocio.

2.1 Procesos de Negocio

Actualmente, la mayoría de las empresas, para conseguir sus objetivos de negocio, organizan su actividad empresarial por medio de *procesos de negocio*. Cada uno de estos procesos se caracteriza por una colección de datos que son producidos y manipulados mediante un conjunto de tareas en las que ciertos agentes (ej., trabajadores o departamentos) participan coordinadamente de acuerdo a un flujo de trabajo determinado [33].

Las nuevas tecnologías de la información permiten que los procesos de negocio sean automatizados, monitorizados, gestionados y optimizados. Es lo que conoce como *Gestión de Procesos de Negocio* (en sus siglas en inglés: BPM). La gestión de los procesos de negocio es una disciplina enfocada en administrar y optimizar de forma continua los procesos de negocio de una empresa.

Los procesos de negocio son capturados y documentados en los *modelos de proceso de negocio.* Estos modelos representan gráficamente los procesos de negocio, especificando sus datos, actividades (o tareas), roles (o agentes) y reglas de negocio. Además, el modelado de procesos facilita el acercamiento y el acuerdo con los clientes, mejora la motivación de los empleados y agiliza la respuesta frente a cambios en el contexto de la empresa [33].

Los modelos de procesos de negocio constituyen el componente fundamental de la gestión de procesos de negocio ya que es el propio modelo el que puede determinar la ejecución del proceso en un motor de ejecución de procesos [26]. Sin embargo, para poder modelar los

procesos, es necesario identificar las tareas, los datos, los roles y las reglas que deben llevarse a cabo. Así, los componentes básicos del modelo de proceso de negocio son:

- *Actividades*: son las tareas o *unidades de trabajo* que deben llevarse a cabo dentro del proceso de negocio. Por ejemplo, en las Figuras 1 y 2: "Identificar pasajero" o "Asignar asiento".
- *Roles*: son los responsables de ejecutar las actividades del proceso. Por ejemplo, en las Figuras 1 y 2: "Maquina de autoservicio" o "Sistema WEB".
- *Flujos de secuencia*: son las secuencias de orden que se define entre las actividades del proceso, eventos y puertas lógicas.
- *Eventos*: son "algo que sucede" antes o durante el proceso de negocio, y afecta el flujo del proceso. Suelen tener una causa o un resultado. Por ejemplo, en las Figuras 1 y 2: "la facturación empieza 23 horas antes de salida".
- *Puertas lógicas*: se emplean para controlar la convergencia y/o divergencia de los flujos de secuencia del proceso. Determinan ramificaciones, bifurcaciones, combinaciones y fusiones del flujo del proceso. Por ejemplo, en las Figuras 1 y 2 se puede ver la puerta lógica AND que determina una bifurcación entre las dos actividades "Dejar equipaje" e "Imprimir tarjeta de embarque para acompañante".
- *Operaciones*: describen las operaciones automáticas (típicamente servicios web) que implementan las actividades del proceso de negocio.

Existen diferentes lenguajes para modelar los procesos de negocio. Concretamente, se destacan (por orden cronológico): IDEF (Integration DEFinition) [21], Diagramas de Actividad UML [22], ebXML BPSS 8 [23] y Business Process Modeling Notation (BPMN) [24], que conforma actualmente el estándar para modelar los procesos de negocio.

La notación BPMN proviene del consorcio *Business Process Management Initiative* (BPMI), la cual fue creada por empresas internacionales tales como Adobe Systems [35], IBM Corporation [34], y SAP AG [36], entre otras. El objetivo de esta iniciativa era la definición de

una notación para procesos de negocio que fuera fácilmente comprendida por los distintos participantes en los mismos (ej. analistas, desarrolladores y clientes). Posteriormente, BPMI fue adoptado por la OMG (Object Management Group [26]) como notación estándar para el modelado de procesos de negocio, unificando y consolidando, de esta manera, las diferentes notaciones existentes.

BPMN proporciona cuatro categorías de elementos de modelado con las cuales es posible representar un proceso:

- Elementos de flujo.
- Elementos de conexión.
- Contenedores y Compartimentos.
- Artefactos.

Elementos de flujo

Se usan para definir el comportamiento del proceso de negocio y existen tres tipos:

- *Evento*: representan algo que ocurre en el proceso. Suelen tener una causa o un resultado y se representan con un círculo. En la Figura 4 se muestran los eventos que se definen en BPMN. Existen varias tipos de eventos:
  - *Simple*: Eventos sin tipo específico. Indican puntos de inicio, de fin y situaciones intermedias.
  - *Mensaje*: Eventos que representan la recepción y envío de mensajes.
  - *Temporal*: Eventos que representan puntos en el tiempo, lapsos, o límites (timeouts).
  - *Escalable*: Eventos que representan un cambio a un nivel más alto de responsabilidad.
  - *Condicional*: Eventos que representan una reacción a cambios en las condiciones de negocios o integración de reglas de negocio.
  - *Error*: Eventos que representan la captura y lanzamiento de errores conocidos con nombre.

- *Cancelación*: Eventos que representan una reacción a la cancelación de una transacción/ Solicitud de cancelación.
- *Compensación*: Eventos que representan el manejo de compensar un fallo parcial de operación.
- *Señal*: Eventos que representan un intercambio de señales entre procesos. Una señal puede ser capturada varias veces.
- *Múltiple*: Eventos que representan la captura uno de un conjunto de eventos. Además, como resultado se lanzan todos los eventos definidos.
- *Paralela Múltiple*: Eventos que representan la captura todos los eventos de un conjunto de eventos en paralelo.
- *Terminación*: Eventos que representan la terminación inmediata del proceso.

Además, existen tres grupos de eventos:

- *Eventos de Inicio:* indican cuando un proceso se inicia.
- *Eventos Intermedios:* indican algo que ocurre o puede ocurrir durante el trascurso de un proceso, entre el inicio y el fin.
- *Eventos de Fin:* indican cuando un proceso acaba.

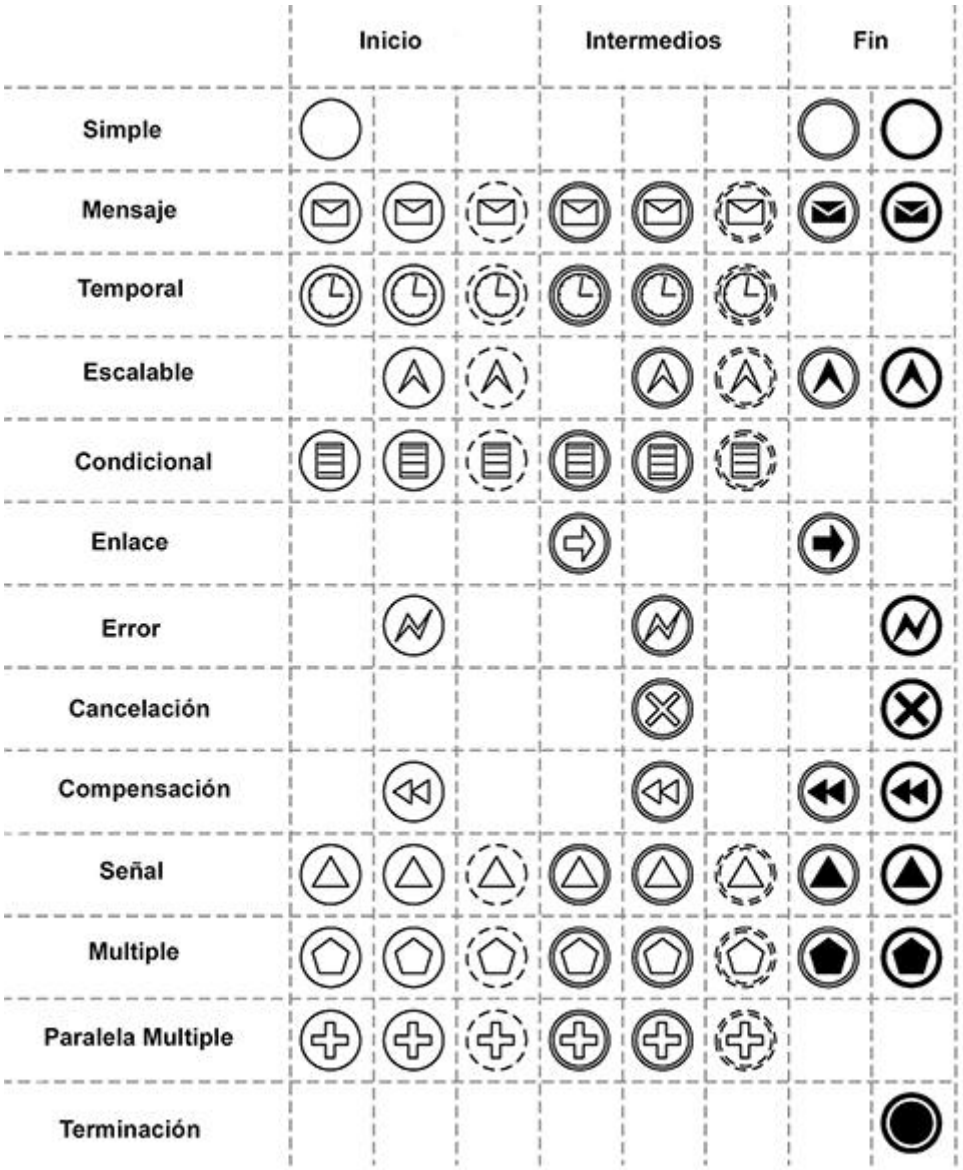


Figura 4: Notación BPMN – Elementos de flujo: Eventos

- *Tarea:* representan una acción o unidad de trabajo realizada por un participante de un proceso. Las tareas pueden ser *atómicas* (tareas individuales) o *complejas* (subproceso), las cuales incluyen un conjunto interno de actividades (otro proceso). Además, se pueden incluir atributos para indicar si la tarea se realiza una sola vez o se repite, de manera cíclica (tarea cíclica) o en paralelo (instancias múltiples). En la Figura 5 se muestran los tipos de tareas definidas en BPMN.

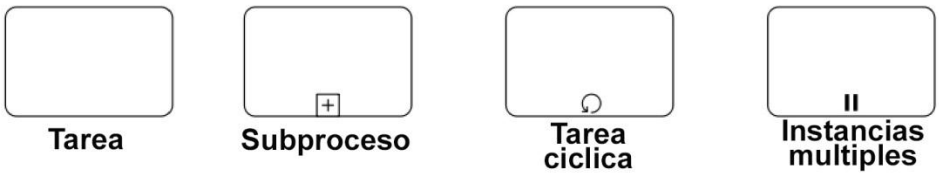


Figura 5: Notación BPMN – Elementos de flujo: Actividades

- *Puertas lógicas o compuertas*: sirven para controlar la convergencia y/o divergencia en la secuencia del flujo de proceso. En la Figura 6 se muestran los tipos de compuertas definidas.

Figura 6: Notación BPMN – Elementos de flujo: Compuertas

  - *Exclusiva*: en un punto de divergencia, la compuerta activa exactamente como flujo saliente un flujo de secuencia de entre las alternativas existentes. En un punto de convergencia, la compuerta espera a que un flujo incidente se complete para activar el flujo saliente.
  - *Basada en eventos*: esta compuerta siempre será seguida por eventos o tareas de recepción y sólo activa un flujo saliente dependiendo del evento que ocurra en primer lugar.
  - *Paralela*: en un punto de divergencia, todos los flujos salientes son activados simultáneamente. En un punto de convergencia, la compuerta espera a que todos los flujos incidentes se completen antes de activar el flujo saliente.
  - *Inclusiva*: en un punto de bifurcación, al menos un flujo saliente es activado. En un punto de convergencia, espera a todos los flujos que fueron activados para activar al saliente.

- *Complejo*: cualquier comportamiento complicado de convergencia/bifurcación no capturado por el resto de compuertas.
- *Exclusiva basada en eventos*: en la ocurrencia de uno de los eventos subsecuentes se crea una nueva instancia del proceso.
- *Paralela basada en eventos:* en la ocurrencia de todos los eventos subsecuentes se crea una nueva instancia del proceso.

Elementos de conexión

Se usan para conectar entre sí elementos de flujo (flujo de secuencia o flujo de mensaje) o elementos de flujo con otros elementos BPMN (ej. contenedores, compartimentos o artefactos). En la Figura 7 se muestran los diferentes tipos de elementos de conexión.

Figura 7: Notación BPMN – Elementos de conexión

- *Flujo de secuencia:* indican el orden en el cual las actividades son ejecutadas en el proceso. Este tipo de elemento se especializa en: normal, condicional y por defecto.
- *Mensajes:* indican el flujo de los mensajes que intervienen entre dos participantes del proceso.

Contenedores y Compartimentos

Los contenedores y los compartimentos representan a las entidades responsables y ejecutoras de las actividades en un proceso. Los compartimentos pueden anidarse en contenedores y, a su vez, en otros

compartimentos. La Figura 8 muestra un ejemplo genérico de estos contenedores y compartimentos.

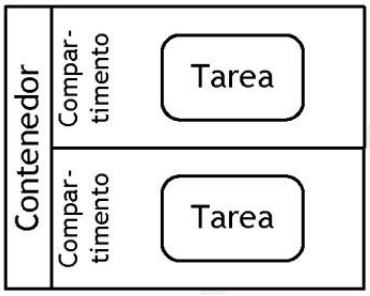


Figura 8: Notación BPMN – Contenedores y Compartimentos

Artefactos

Son elementos que no influyen en el cambio del flujo de ejecución pero mejoran la comprensión del modelo del proceso de negocio. En la Figura 9 se representan estos elementos.

Figura 9: Notación BPMN – Artefactos

- *Grupo*: permite agrupar actividades con el único propósito de documentar las actividades que, de alguna manera, están relacionadas entre sí.
- *Objeto de datos*: representa la información que una actividad necesita para ejecutarse o provee como resultado.
- *Anotaciones*: permite añadir información para mejorar el entendimiento del modelo.

2.2 Ciclo de Vida de los Procesos de Negocio

Los procesos de negocio se engloban en una serie de fases que se relacionan entre sí en una estructura cíclica (ver Figura 10). A continuación se detallan cada una de estas fases centrándose, para cada una de ellas, en los aspectos clave de la variabilidad en los procesos de negocio.

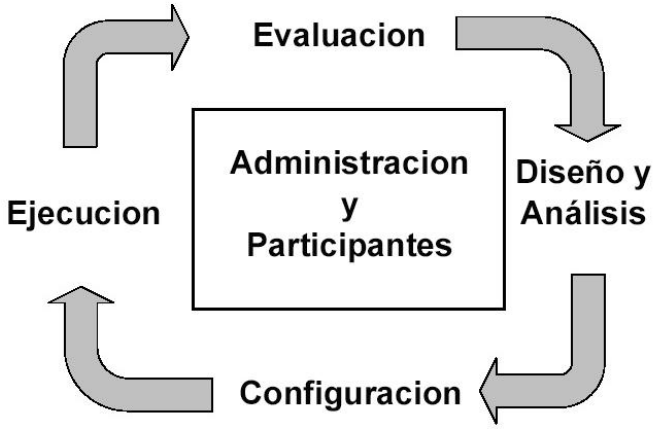


Figura 10: Ciclo de vida de los procesos de negocio

Diseño y Análisis

Típicamente, el ciclo de vida de los procesos de negocio comienza en la fase de *diseño y análisis*. Esta fase se centra en identificar los procesos de negocio y su entorno organizativo y técnico. Basándose en esta identificación, los procesos de negocio se capturan en *los modelos de procesos de negocio*, los cuales se expresan en una notación gráfica concreta y facilitan la comunicación entre los participantes de los procesos. La mayoría de los sistemas de gestión de procesos de negocio proporcionan un entorno de modelado que se puede utilizar en esta fase (WebSphere Business Modeler Advanced [19], Bizagi Process Modeler [20]). En el contexto de la variabilidad, durante esta fase, las posibles variantes de un proceso se definen en términos de un modelo de proceso configurable, que representa una familia de procesos completa y elimina redundancias mediante la representación de las partes comunes sólo una vez (ver Figura 11). Una vez definido el modelo de proceso configurable, éste debe ser validado (en términos de su contenido) y verificado (en

términos de su construcción). Por ejemplo, un instrumento útil para validar un modelo de proceso configurable podría ser un taller (o workshop), durante el cual las personas involucradas discuten las variantes del proceso y el modelo que las representa [28]. Concretamente, los participantes del taller comprueban si todos los requisitos de negocio se reflejan en el modelo de proceso configurable. Además, técnicas de simulación pueden utilizarse para apoyar la validación, ya que pueden representar ciertas secuencias de ejecución no deseadas y mostrar deficiencias en el modelo de proceso configurable.

Configuración

Durante esta fase, el sistema debe configurarse de acuerdo con el entorno de la organización de la empresa [27]. Esta configuración incluye las interacciones de los participantes con el sistema, así como la integración de los sistemas de software existentes con el sistema de gestión de procesos de negocio. En el contexto de la variabilidad de procesos, en esta fase, se realiza un proceso de *individualización* y otro de *selección* con el fin de derivar una variante del proceso para un contexto de aplicación concreto (ver Figura 11). Primeramente, en el paso de individualización se identifican todas las posibles variantes de modelo del proceso para todos los contextos. A continuación, en el paso de selección se escoge una variante del proceso concreta que cumpla los requisitos del negocio y que es válida para un contexto de aplicación determinado.

Ejecución

Una vez que la fase de configuración del sistema se completa, las representaciones ejecutables de los modelos de procesos de negocio (o las variantes del proceso en el contexto de la variabilidad), llamados *instancias de proceso,* pasan a ser ejecutados. Concretamente, cada instancia se inicia a raíz, típicamente, de un evento predefinido, como por ejemplo, el requerimiento de facturar un billete de avión. Normalmente, el sistema de gestión de procesos de negocio se encarga de controlar la ejecución de las instancias.

Durante esta fase, es recomendable el uso de técnicas de monitorización que permitan visualizar y controlar el estado de la

ejecución de las instancias. El control de estas instancias es un mecanismo clave para informar de forma precisa y constante sobre el estado de la ejecución del proceso de negocio. Estas técnicas de monitorización, por lo general, almacenan los datos de la ejecución en algún tipo de archivo de registro (ej. un documento de texto, una base de datos). Estos archivos contienen los conjuntos de registros ordenados de todos los eventos que han ocurrido durante la ejecución de la instancia. Así, esta información del registro puede ser utilizada, posteriormente, para evaluar los procesos en la siguiente fase del ciclo de vida.

Evaluación

En esta última fase se utiliza la información capturada durante la ejecución del proceso para evaluar y mejorar los modelos de procesos de negocio y sus implementaciones. Los registros de ejecución obtenidos de la fase anterior se evalúan para identificar la calidad de los actuales modelos de procesos de negocio (o variantes) y la adecuación a su entorno de ejecución. Por ejemplo, se puede identificar que una determinada actividad tarda demasiado tiempo por la falta de los recursos necesarios para su realización.

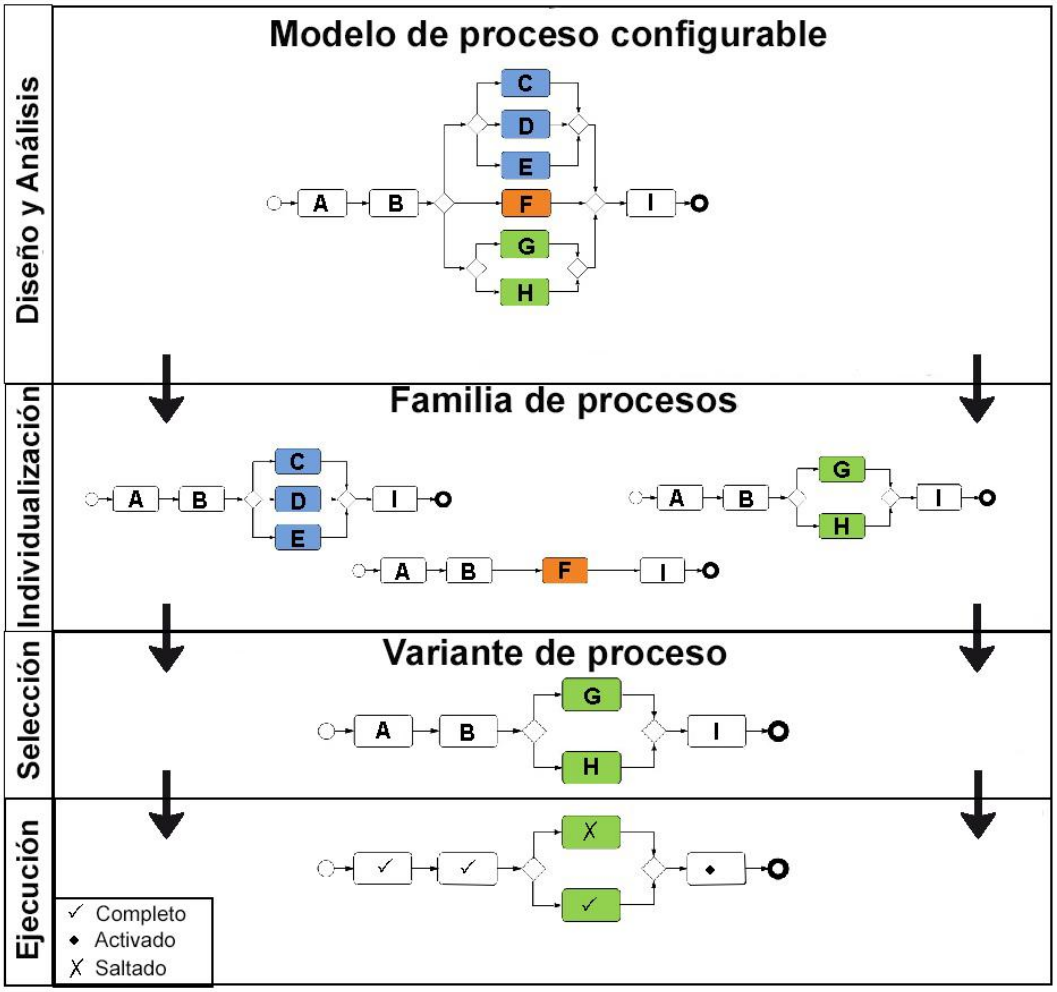


Figura 11: Transición desde el análisis y diseño de una familia de procesos a ejecución de una instancia de una variante concreta

3 VARIABILIDAD EN LOS PROCESOS DE NEGOCIO

Como se ha descrito en la Sección 1 de este trabajo, gestionar la variabilidad en los procesos de negocio resulta fundamental para poder manejar de forma eficiente la cantidad de variantes de procesos de las que disponen actualmente las empresas. Normalmente, para un proceso particular existe una multitud de variantes de proceso cada una de los cuales es válida en un contexto particular. Por ejemplo, en cuanto a la gestión de producción de los componentes eléctricos en un vehículo [10], se han identificado más de veinte variantes del proceso en función de la serie de productos, proveedores, o fases de desarrollo de los vehículos. Para ilustrar esta variabilidad, se usa el proceso que gestiona la reparación del vehículo en un garaje extraído de [2] (ver Figura 12a) El proceso se inicia con la recepción de un vehículo. Después de realizar el diagnóstico del mismo, el vehículo se repara si es necesario. Durante el diagnóstico y la reparación se hacen en paralelo trabajos de mantenimiento, por ejemplo, se comprueban el aceite y el líquido del limpiaparabrisas y se reponen si es necesario. El proceso termina al entregar el vehículo reparado y listo para el cliente. Según el contexto de proceso (ej. país, garaje o tipo de vehículo) se requieren diferentes variantes de este proceso. Las Figuras 12b-12d muestran tres variantes (simplificadas) de este proceso de reparación de vehículos. En la variante 1 (ver Figura 12b) se asume que el vehículo dañado requiere de una lista de “Tipo 2” para realizar el diagnóstico. Por lo tanto, las actividades “Diagnóstico” y “Reparación” se adaptan modificando su atributo "Lista de comprobaciones" al valor “Tipo 2”. Además, el garaje omite el mantenimiento del vehículo, ya que se considera como servicio especial que no se ofrece conjuntamente con el proceso de reparación. A nivel de modelado, esto se plasma mediante la omisión de la actividad “Mantenimiento”. Como un segundo ejemplo, se considera variante 2 (ver Figura 12c). En este caso, debido a la legislación específica de cada país, se requiere un control de seguridad final antes de entregar el vehículo al cliente. Con respecto a esta variante, una nueva actividad “Verificación final” tiene que ser añadida. Por último, la Variante 3 (ver Figura 12d) considera una lista de “Tipo 2” para el diagnóstico y reparación de vehículos, donde el garaje no ofrece mantenimiento para el proceso de

reparación, y hay normas legales que requieren un control de seguridad final.

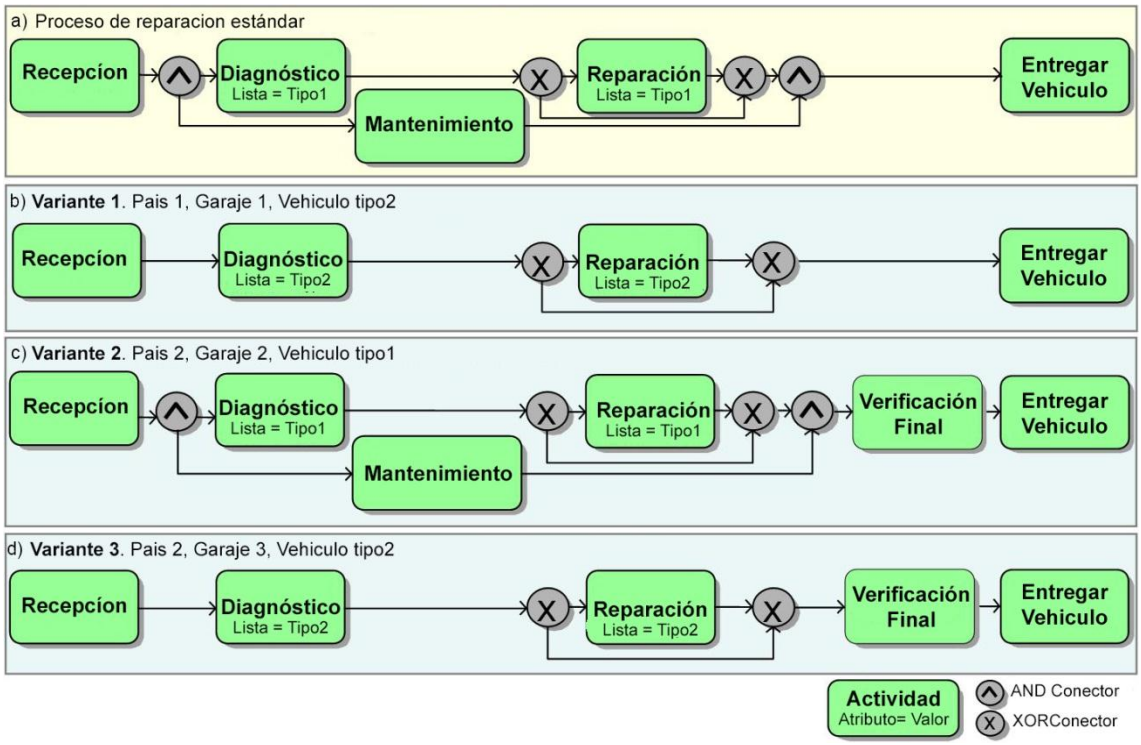


Figura 12: Las variantes de un proceso de reparación de vehículos

Manejar de forma eficiente familias de procesos como la del ejemplo anterior requiere el uso de construcciones específicas propias de la variabilidad [4]. Sin embargo, la introducción de estas construcciones implica una complejidad adicional en relación con el lenguaje de modelado utilizado. Para que estas propuestas puedan usarse de manera industrial, la calidad de los modelos de procesos configurables creados es crucial y, por tanto, se necesita un apoyo eficaz a los diseñadores de procesos cuando manejan familias de proceso. Para ello, en [4] se han identificado dichas construcciones (ver Sección 3.1) y un conjunto de patrones de cambio para poder manejarlas (ver Sección 3.2)

3.1 Construcciones del Lenguaje para Manejar la Variabilidad

Gestionar la variabilidad en los procesos de negocio implica una complejidad adicional que debe ser gestionada de manera diferente [4]. Entre otros retos, supone el tratamiento de nuevas construcciones del lenguaje que permiten representar de forma explícita la variabilidad en los

modelos de procesos de negocio. En [4], a través de una búsqueda sistemática de las construcciones del lenguaje más utilizadas en la literatura existente sobre variabilidad en procesos de negocio, se han identificado qué construcciones a nivel de lenguaje (de alto nivel de abstracción) son necesarias para representar dicha variabilidad. A continuación se describen estas construcciones en detalle.

Construcción 1 (*Región Configurable*): se define como una región (punto concreto) en un modelo de proceso configurable para la que pueden existir diferentes opciones de configuración en función del contexto de aplicación. Por ejemplo, en el contexto del proceso de facturación de un billete de avión (ver Ejemplo 1), una aerolínea ofrece diferentes formas de obtener las tarjetas de embarque según el tipo de facturación: (1) imprimir una tarjeta de embarque en el mostrador de la aerolínea, (2) imprimir una tarjeta de embarque en casa, o (3) su obtención a través de teléfono móvil. El punto concreto del modelo de proceso configurable donde se deben definir estas alternativas constituye una *Región Configurable.*

Construcción 2 (*Alternativa de Configuración*): se define como una opción de configuración que puede ser seleccionada para una *Región Configurable* específica. Por ejemplo, existen diferentes tipos de tarjeta de embarque resultantes de facturar un billete de avión: de papel, electrónica, o en el teléfono móvil. Cada una de estos tipos de tarjeta de embarque constituye una *Alternativa de Configuración.*

Construcción 3 (*Condición de Contexto*): se define como las condiciones del entorno (contexto) en las que se puede seleccionar una *Alternativa de Configuración* para una *Región Configurable* específica. Por ejemplo, el hecho de que los pasajeros con equipaje con sobrepeso paguen un cargo adicional supone una *Condición de Contexto* para ejecutar el propio pago.

Construcción 4 (*Restricción de Configuración*): se define como una restricción en la selección de las *Alternativas de Configuración.* Estas restricciones se basan en condiciones semánticas que garantizan el uso adecuado de las *Alternativas de Configuración*. Por ejemplo, la necesidad

de localizar personal adicional cuando pasajeros con alguna discapacidad viajan supone una *Restricción de Configuración*

3.2 Patrones de Cambio Para Familias de Procesos

En esta Sección se presentan 9 patrones de cambio identificados como relevantes para gestionar las familias de proceso [4]. Estos patrones se basan en las cuatro construcciones del lenguaje presentadas en la Sección anterior. Los *patrones de cambio para familias de* procesos son genéricos y de alto nivel de abstracción ya que abstraen detalles específicos de cada lenguaje de modelado de familias de procesos, pudiéndose aplicar con cualquiera de ellos (es decir, son independientes del lenguaje). Además, tienen la intención de ser completos ya que cubren todos los cambios relacionados con las construcciones del lenguaje propias para gestionar la variabilidad en procesos de negocio descritas en la Sección anterior. Los patrones de cambio se dividen en tres categorías: insertar, eliminar y modificar dichas construcciones (ver Tabla 1).

CP1: INSERTAR Región Configurable
CP2: BORRAR Región Configurable
CP3: INSERTAR Alternativa de Configuración
CP4: BORRAR Alternativa de Configuración
CP5: INSERTAR Condición de Contexto
CP6: BORRAR Condición de Contexto
CP7: MODIFICAR Condición de Contexto
CP8: INSERTAR Restricción de Configuración
CP9: BORRAR Restricción de Configuración

Tabla 1: Patrones de cambio para familias de proceso.

A continuación se detallan estos patrones indicando, para cada uno de ellos, una descripción, un ejemplo dentro del contexto del proceso de facturación de un billete de avión, una descripción del problema abordado, las posibles opciones de diseño correspondientes (si hay más de una) y la forma de implementarlos de manera genérica.

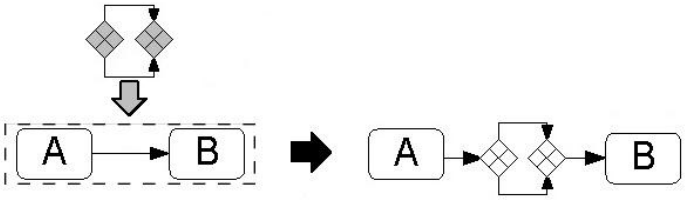
Patrón CP1. INSERTAR Región Configurable
Descripción: En un modelo de proceso configurable, se añadirá una *Región Configurable*.
Ejemplo: La forma de administrar las tarjetas de embarque depende del tipo de facturación a realizar (ej., tarjetas de embarque electrónicas y en papel). Se asume que en el modelo de proceso configurable original todavía no ha considerado estas variaciones. Por lo tanto, es necesario añadir una Región Configurable en el modelo de proceso configurable para reflejar esta variabilidad.
Problema: En una cierta posición en el modelo de proceso configurable, diferentes Alternativas de Configuración existen y no se reflejan en el modelo de proceso configurable todavía. Por lo tanto, es necesario añadir una Región Configurable que permite definir estas Alternativas de Configuración.
Implementación: Se insertan dos puertas XOR conectadas entre sí junto con dos flujos de secuencia que conectan dichas puertas con los dos elementos seleccionados. Estas dos puertas XOR representan los límites de entrada y salida de la Región Configurable. 

Tabla 2: Descripción del patrón CP1

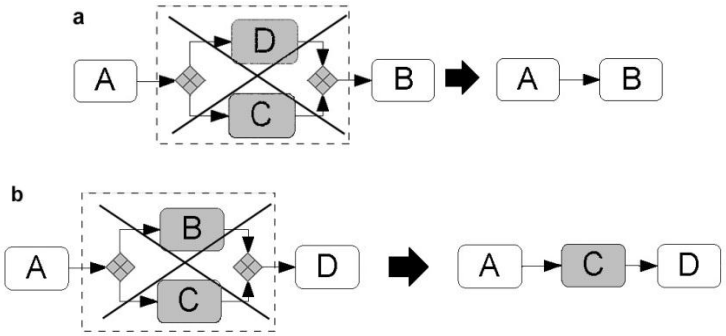
Patrón CP2. BORRAR Región Configurable
Descripción: En un modelo de proceso configurable, se borrará una *Región Configurable*.
Ejemplo: Se asume que en el modelo de proceso configurable original existe una *Región Configurable* que captura la variabilidad para la obtención de una tarjeta de embarque. Sin embargo, con el fin de ahorrar dinero, la aerolínea ahora sólo ofrece la tarjeta de embarque electrónica (es decir, otras Alternativas de Configuración ya no se ofrecen y por lo tanto la Región Configurable ya no es necesaria.
Problema: Una *Región Configurable* ya no es necesaria y por lo tanto debe ser borrada.
Opciones del diseño: Existen dos opciones de diseño diferentes: **1.** Borrar la Región Configurable eliminando todas las Alternativas de Configuración que contiene dicha Región Configurable. **2.** Borrar la Región Configurable manteniendo exactamente una de las Alternativas de Configuración existentes (es decir, la Alternativa de Configuración pasa a ser una parte común a todas las variantes del proceso).
Implementación: Se borran dos puertas XOR (ejemplo a) o se borran dos puertas y todas Alternativas de Configuración menos una (ejemplo b) 

Tabla 3: Descripción del patrón CP2

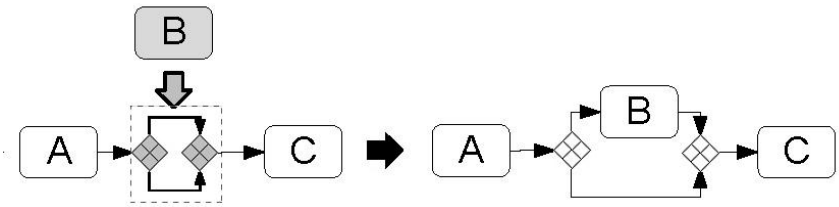
Patrón CP3. INSERTAR Alternativa de Configuración
Descripción: En un modelo de proceso configurable, se añadirá una Alternativa de Configuración en una Región Configurable.
Ejemplo: Se asume que existe una *Región Configurable* en el modelo de proceso configurable original que captura la variabilidad para la obtención de la tarjeta de embarque. Se asume, además, que la aerolínea ahora quiere ofrecer la posibilidad de obtener también tarjetas de embarque en los teléfonos móviles. Así, es necesario añadir una nueva alternativa a esta *Región Configurable* que refleje esta nueva opción.
Problema: Para una *Región Configurable* específica del modelo de proceso configurable, las Alternativas de Configuración existentes no cubren todas las posibles opciones de configuración y, por tanto, es necesario añadir una Alternativa de Configuración adicional.
Implementación: Se inserta una actividad *entre dos puertas configurables XOR*. Esta actividad es la que representa la Alternativa de Configuración a insertar. 

Tabla 4: Descripción del patrón CP3

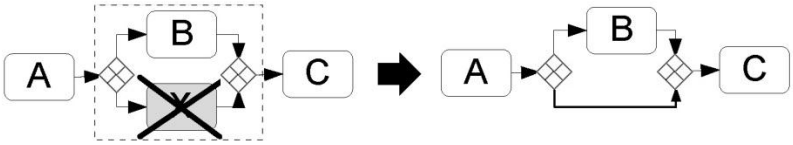
Patrón CP4. BORRAR Alternativa de Configuración
Descripción: En un modelo de proceso configurable, se borrará una Alternativa de Configuración de una *Región Configurable* específica.
Ejemplo: Se asume que existe una Región Configurable en el modelo de proceso configurable que captura la variabilidad para la obtención de las tarjetas de embarque. Se asume, además, que por razones económicas, la aerolínea no ofrece la tarjeta de embarque en papel nunca más y sólo permite las tarjetas electrónicas y en teléfonos móviles. Por lo tanto, la Alternativa de Configuración de la impresión de la tarjeta de embarque de papel ya no es necesaria.
Problema: Una Alternativa de Configuración ya no es necesaria y por lo tanto debe ser borrada.
Implementación: Se borra una actividad entre dos puertas configurables XOR (que delimitan la Región Configurable). 

Tabla 5: Descripción del patrón CP4

Patrón CP5. INSERTAR Condición del Contexto
Descripción: En un modelo de proceso configurable, se añadirá una condición de contexto relacionada a una Alternativa de Configuración de una Región Configurable para definir cuándo se seleccionará la Alternativa de Configuración para dicha región.
Ejemplo: Un pasajero que lleva equipaje superior a 20kg deberá abonar un cargo adicional.
Problema: Una condición de contexto se añade a un modelo de proceso configurable para especificar las condiciones bajo las cuales se seleccionará una Alternativa de Configuración en una Región Configurable específica.

Tabla 6: Descripción del patrón CP5

Patrón CP6. BORRAR Condición del Contexto
Descripción: En un modelo de proceso configurable, se borrará una condición del contexto de una Alternativa de Configuración.
Ejemplo: Los pasajeros VIP originariamente no tenían que pagar un cargo adicional por el sobrepeso de equipaje. Sin embargo, la aerolínea decide que a partir de ahora todos los pasajeros deben pagar dicho cargo.
Problema: Una condición de contexto ya no es necesaria para la selección de una Alternativa de Configuración de una *Región Configurable* y por lo tanto se suprime.

Tabla 7: Descripción del patrón CP6

Patrón CP7. MODIFICAR Condición del Contexto
Descripción: En un modelo de proceso configurable, se modificará una condición de contexto.
Ejemplo: El pago de un cargo adicional es necesario cuando el peso del equipaje es superior de 20kg. Debido a nuevas directrices de la aerolínea, la cuota adicional ahora es necesaria cuando el equipaje pesa más de 15 kg.
Problema: Una condición de contexto ya no es adecuada y se modificará en el modelo de proceso configurable.

Tabla 8: Descripción del patrón CP7

Patrón CP8. INSERTAR Restricción de Configuración
Descripción: En un modelo de proceso configurable, se añadirá una restricción sobre el uso de varias Alternativas de Configuración.
Ejemplo: Cuando viajan pasajeros discapacitados, se requiere personal de la aerolínea adicional para que lo acompañe hasta a la puerta de embarque. Es decir, existe una restricción (de inclusión) entre que viaje un pasajero discapacitado y la necesidad de personal adicional que lo acompañe.
Problema: El uso de Alternativas de Configuración necesita ser condicionado en un modelo de proceso configurable y por lo tanto, se añade una Restricción de Configuración.
Implementación: Se inserta una Restricción de Configuración (representada por un rectángulo) vinculada a las puertas XOR que delimitan las *Regiones Configurables* a las cuales pertenecen las actividades que representan las Alternativas de Configuración afectadas por la Restricción de

Configuración.

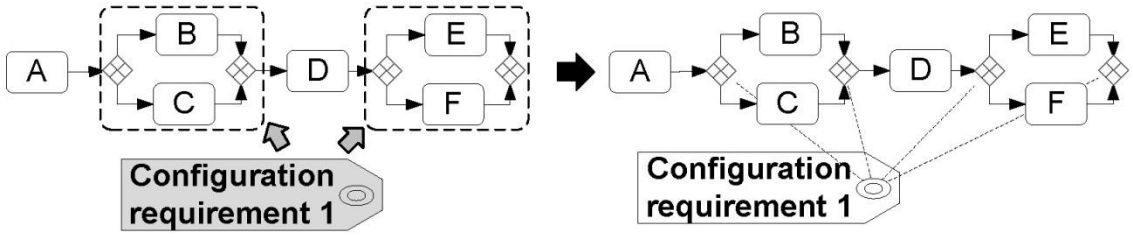


Tabla 9: Descripción del patrón CP8

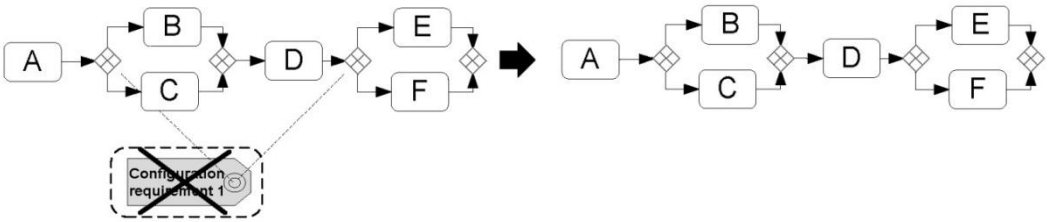
Patrón CP9. BORRAR Restricción de Configuración
Descripción: En un modelo de proceso configurable, se borrará una restricción entre varias Alternativas de Configuración.
Ejemplo: Cuando viaja un pasajero discapacitado, se requiere personal de la aerolínea que lo acompañe hasta la puerta de embarque (restricción de inclusión). Debido a nuevas regulaciones legales, de ahora en adelante sólo sus familiares podrán acompañarlos hasta la puerta de embarque en lugar del personal de la aerolínea. Es decir, ya no es necesaria la restricción de inclusión entre que viaje un pasajero discapacitado y la necesidad de personal adicional de la aerolínea.
Problema: Una restricción entre dos o más Alternativas de Configuración ya no es necesaria y por lo tanto debe ser borrada.
Implementación: Se elimina una Restricción de Configuración (representada por un rectángulo) vinculada a las puertas lógicas XOR las cuales delimitan las regiones configurables a las que pertenecen las actividades que representan las Alternativas de Configuración afectadas por la Restricción de Configuración. 

Tabla 10: Descripción del Patrón CP9

4 DESARROLLO DE LA PROPUESTA

En esta Sección se describirá el proceso de desarrollo del trabajo. Primero se describe el ámbito tecnológico. Y finalmente se describe la implementación realizada.

Como CEP sobre la que se realiza la implementación de los patrones sólo soporta BPMN sin ningún tipo de soporte para la variabilidad, se asume que, para representar Regiones Configurables, se utilizarán puertas lógicas XOR que tengan enlazada una Restricción de Configuración. Además, como BPMN tampoco ofrece mecanismos para representar las Condiciones del Contexto que permitan especificar el contexto de cada Alternativa de Configuración, no se han implementado los patrones referentes a ellas (patrones CP5, CP6, CP7).

Así, se han implementado los siguientes patrones de cambio:

- CP1: Insertar Región Configurable.
- CP2: Borrar Región Configurable.
- CP3: Insertar Alternativa de Configuración.
- CP4: Borrar Alternativa de Configuración.
- CP8: Insertar Restricción de Configuración.
- CP9. Borrar Restricción de Configuración.

Para describir la implementación realizada, en la Sección 4.1 se presenta el ámbito tecnológico donde se describe el lenguaje de programación que subyace en CEP y la plataforma tecnológica sobre la que CEP está implementado. Además, en la Sección 4.2 se presenta la implementación de los patrones, cómo es posible configurar una variante del proceso en base a una Restricción de Configuración.

4.1 Ámbito Tecnológico

La implementación de este trabajo se ha llevado a cabo en lenguaje Java sobre la plataforma Eclipse, versión Kepler Service Release 1.

CEP es un archivo ejecutable de Eclipse que puede ser utilizado en cualquier sistema operativo pero con una capacidad de memoria determinada.

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems [37] a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple eliminando herramientas de bajo nivel (que suelen inducir a muchos errores) como, por ejemplo, la manipulación directa de punteros o memoria [9].

Eclipse es una plataforma de desarrollo diseñada para ser extendida de forma indefinida a través de *plug-ins* [11]. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de distintas herramientas de desarrollo. Además, no está ligada un lenguaje específico sino que es un entorno de desarrollo (IDE, Integrated Development Environment) genérico. Igualmente, proporciona herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones.

4.2 Descripción Detallada de la Implementación Realizada

En esta Sección se presenta cómo se implementan los patrones de cambio y, a continuación, cómo ha sido realizada esta implementación en CEP.

4.2.1 Implementación de los Patrones

En este apartado se describe cómo se han implementado los patrones de cambio CP1 (*Insertar Región Configurable*), CP2 (*Borrar Región Configurable*), CP3 (*Insertar Alternativa de Configuración en Región Configurable*), CP4 (*Borrar Alternativa de Configuración en Región Configurable*), CP8 (*Insertar Restricción de Configuración entre unas Regiones Configurables*), CP9 (*Borrar Restricción de Configuración entre unas Regiones Configurables*).

Los patrones de cambio se aplican mediante un panel de patrones en CEP donde están accesibles todos los patrones que pueden ser utilizados en CEP. En la Figura 13 se puede observar este panel.

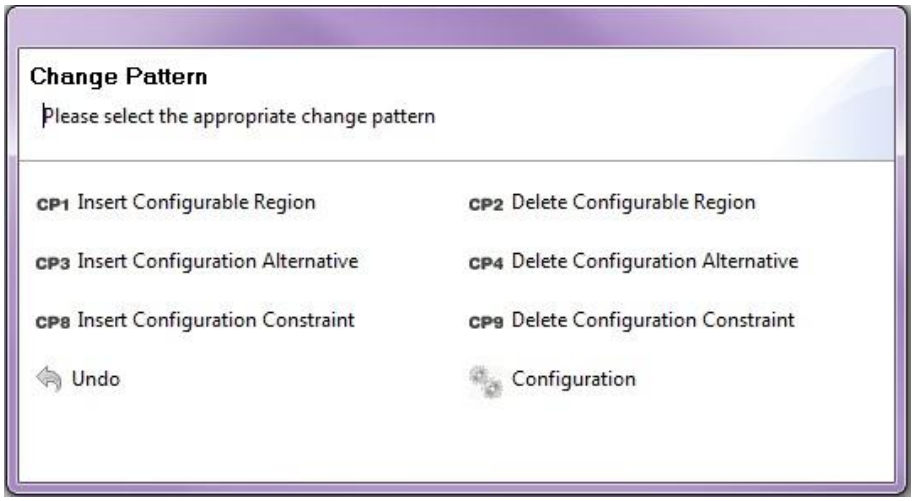


Figura 13: Panel de patrones de cambio

A continuación, se describe, para cada patrón: su descripción, condiciones de aplicación, algoritmo de aplicación, acciones que realiza el patrón e instrucciones de aplicación del patrón de cambio.

Patrón CP1. INSERTAR Región Configurable
Descripción: Se inserta una Región Configurable
Condiciones de aplicación: Se asume que este patrón se aplica siempre y solamente entre dos elementos del modelo de proceso configurable secuenciales conectados mediante un flujo de secuencia. Estos elementos puedes ser de varios tipos, por ejemplo, dos actividades, dos puertas lógicas, o dos eventos.
Acción: Para implementar este patrón en CEP, se utilizan dos puertas lógicas XOR que se insertan entre los dos elementos seleccionados. Además, estas puertas XOR quedan conectadas entre sí mediante un flujo de secuencia. Así, la Región Configurable queda delimitada por dichas puertas XOR.
Instrucciones de aplicación: Para aplicar este patrón hay que seleccionar dos elementos secuenciales y conectados. Si los elementos han sido seleccionados correctamente, en el panel de patrones de CEP se habilita el botón del patrón CP1. A continuación, basta con apretarlo para que se aplique automáticamente el patrón. Si el botón del patrón esta deshabilitado entonces este patrón no se puede aplicar ya que los elementos seleccionados no son correctos (ej. no están secuencialmente conectados). **Algoritmo Habilitar Botón de Aplicación del Patrón CP1** Algoritmo: SI número de elementos del Área seleccionada = 2 ENTONCES

SI los elementos están conectados secuencialmente ENTONCES
Habilitar el botón CP1
FINSI
FINSI
FIN

Algoritmo Implementar Patrón CP1
Algoritmo:
Crear XOR “split”
Crear XOR “join”
Conectar primer elemento del Área seleccionada con XOR “split”
Conectar XOR “split” con XOR “join”
Conectar XOR “split” con XOR “join””
Conectar XOR “join” con segundo elemento del Área seleccionada
FIN
En la Figura se pueden observar algunos ejemplos de cómo actúa la aplicación de este patrón. En el ejemplo 1 (a) este patrón se aplica entre dos eventos (inicio y fin) y se insertan dos puertas lógicas XOR entre ellos. En el ejemplo 2 (b), el patrón CP1 se aplica entre dos actividades y por último en el ejemplo 3 (c) este mismo patrón se aplica entre dos puertas lógicas existentes.

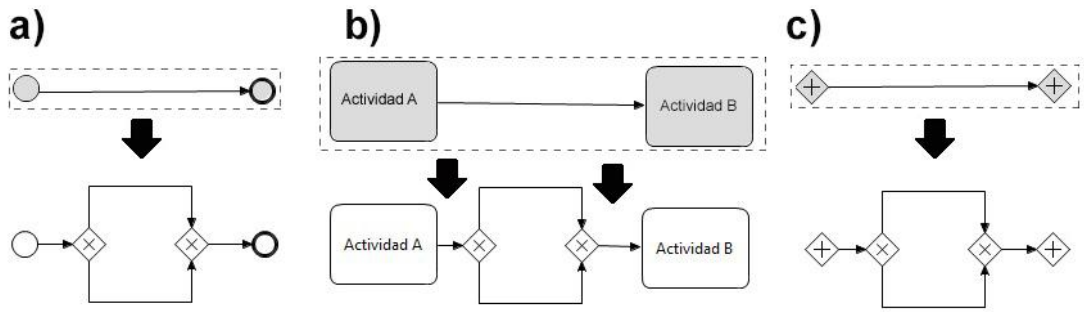


Tabla 11: Implementación de patrón de cambio CP1

Patrón CP2. BORRAR Región Configurable
Descripción: Este patrón borra una Región Configurable del modelo de procesos configurable.
Condiciones de aplicación: Se asume que este patrón se aplica siempre y solamente a una Región Configurable existente en el modelo de proceso configurable.
Acción: Para implementar este patrón en CEP, se soporta la posibilidad de borrar una Región Configurable aunque ésta contenga alternativas (actividades) dentro (ver Figura a). En caso de que la Región Configurable

contenga varias actividades (alternativas) dentro, al aplicar el patrón CP2 el usuario puede elegir: (1) borrar toda la Región Configurable incluyendo todas las alternativas que ésta incluye, o (2) borrar las puertas lógicas y todas las alternativas menos una (ver Figura b).

Instrucciones de aplicación: Para aplicar este patrón hay que seleccionar dos puertas lógicas XOR que representan una Región Configurable. Si los elementos han sido seleccionados correctamente se habilita el botón del patrón CP2 en el panel de patrones de CEP. A continuación, basta con apretarlo para que se aplique el patrón. Si el botón del patrón esta deshabilitado entonces este patrón no se puede aplicar ya que los elementos seleccionados no son correctos. Para poder elegir la opción de cómo borrar la Región Configurable, se muestra al usuario un nuevo diálogo.

Algoritmo: Habilitar Botón de Aplicación del Patrón CP2

Algoritmo:

```
SI número de elementos del Área seleccionada = 2 ENTONCES
   SI Área seleccionada tiene una conexión entrante y una saliente ENTONCES
      SI el primer elemento del Área seleccionada es XOR AND el segundo elemento del Área seleccionada es XOR ENTONCES
            Habilitar el botón CP2
      FINSI
   FINSI
FINSI
FIN
```

Algoritmo: Implementar Patrón CP2

Algoritmo:

```
OPCION "Elige opción del diálogo" DE
   PARA "Borrar región configurable" HACER
      PARA todos elementos de Área seleccionada HACER
         Borrar conexión
         Borrar nodo
      FINHACER
      Conectar elemento anterior del Área seleccionada con elemento posterior
   FINHACER
   PARA "Dejar solo actividad X" HACER
      PARA todos elementos de Área seleccionada HACER
         SI nombre != nombre de actividad X ENTONCES
            Borrar este elemento de Área seleccionada
```

FINSI
FINHACER
Conectar elemento anterior del Área seleccionada con Actividad X
Conectar Actividad X con elemento posterior del Área seleccionada
FINHACER
FINOPCION
FIN

En las Figuras a y b se pueden observar ejemplos de cómo actúa la aplicación de este patrón. En el ejemplo 1 (ver Figura a) este patrón borra toda la Región Configurable incluyendo todas las alternativas que ésta incluye. En el ejemplo 2 (ver Figura b), el patrón CP2 borra las puertas lógicas y todas las alternativas menos una.

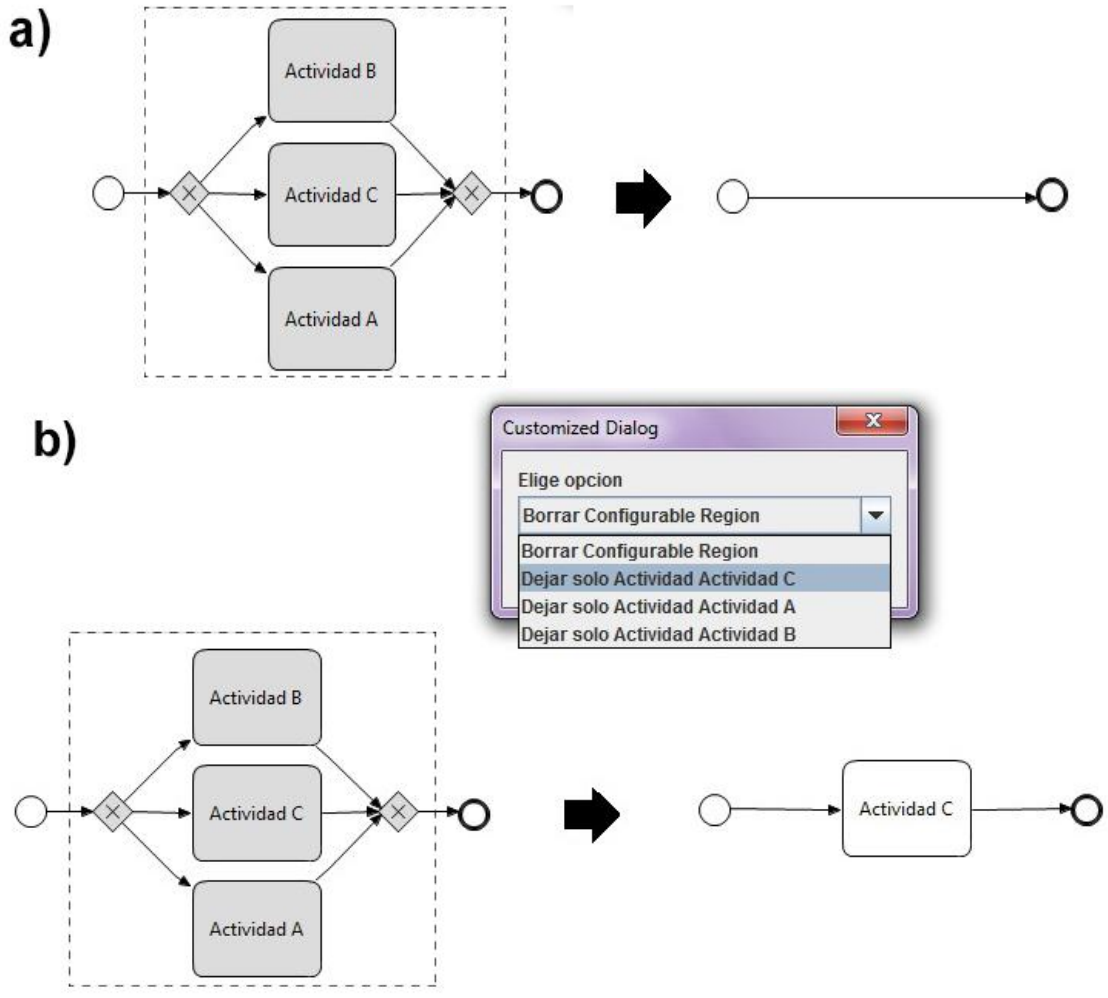


Tabla 12: Implementación de patrón de cambio CP2

Patrón CP3. INSERTAR Alternativa de Configuración

Descripción: Este patrón inserta una Alternativa de Configuración en una región configurable.

Condiciones de aplicación: Se asume que este patrón se aplica siempre y solamente en una Región Configurable existente en el modelo de proceso configurable.

Acción: Para implementar este patrón en CEP, se inserta una actividad (que representa la Alternativa de Configuración) entre las puertas lógicas XOR (que delimitan la Región Configurable). En una Región Configurable, se pueden insertar tantas alternativas como sea necesario.

Instrucciones de aplicación: Para aplicar este patrón hay que seleccionar las dos puertas lógicas XOR que representan una Región Configurable. Si los elementos han sido seleccionados correctamente se habilita el botón del patrón CP3 en el panel de patrones de CEP. A continuación, basta con apretarlo para que se aplique automáticamente el patrón. Si el botón del patrón esta deshabilitado, este patrón no se puede aplicar ya que los elementos seleccionados no son correctos.

Algoritmo: Habilitar Botón de Aplicación del Patrón CP3

```
Algoritmo:
  SI número de elementos de Área seleccionada >= 2 ENTONCES
    SI Área seleccionada tiene una conexión entrante y una saliente ENTONCES
      SI primer elemento del Área seleccionada es XOR AND último elemento del Área seleccionada es XOR ENTONCES
        SI último elemento del Área seleccionada es “join” del “split” de primer elemento del Área seleccionada
          Habilitar el botón
        FINSI
      FINSI
    FINSI
  FINSI
FIN
```

Algoritmo: Implementar Patrón de cambio CP3

```
Algoritmo:
    Crear elemento Actividad a insertar
    Conectar primer elemento del Área seleccionada con Actividad
    Conectar Actividad con segundo  elemento del Área seleccionada
FIN
```

En la Figura se puede observar un ejemplo de cómo actúa la aplicación de este patrón. En este ejemplo, el patrón CP3 inserta la Actividad A en una

Región Configurable delimitada por dos puertas XOR configurables.

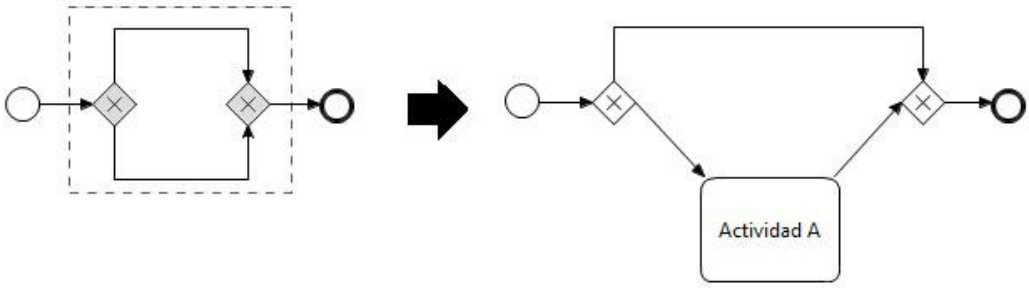


Tabla 13: Implementación de patrón de cambio CP3

Patrón CP4. BORRAR Alternativa de Configuración
Descripción: Este patrón borra una Alternativa de Configuración.
Condiciones de aplicación: Se asume que este patrón se aplica solamente cuando una actividad (que representa la Alternativa de Configuración) está entre dos puertas lógicas XOR (que representan la Región Configurable).
Acción: Para implementar este patrón en CEP, se borra la actividad seleccionada e inserta en su lugar un flujo de secuencia conectando ambas puertas lógicas XOR.
Instrucciones de aplicación: Para aplicar este patrón hay que seleccionar una actividad que representa a una Alternativa de Configuración (entre dos puertas lógicas configurables XOR). Si el elemento ha sido seleccionado correctamente se habilita el botón del patrón CP4 en el panel de patrones de CEP. A continuación, basta con apretarlo para que se aplique automáticamente el patrón. Si el botón del patrón esta deshabilitado, este patrón no se puede aplicar ya que los elementos seleccionados no son correctos. **Algoritmo: Habilitar Botón de Aplicación del Patrón CP4** Algoritmo: SI número de elementos de Área seleccionada = 1 ENTONCES SI elemento de Área seleccionada es Alternativa de configuración ENTONCES Habilitar el botón FINSI FINSI FIN

Algoritmo: Implementar Patrón de cambio CP4

Algoritmo:

Borrar el nodo del Área seleccionada

Conectar elemento anterior del Área seleccionada con elemento posterior

FIN

En la Figura se puede observar un ejemplo de cómo actúa la aplicación de este patrón. En este ejemplo, el patrón CP4 borra la Actividad B de una Región Configurable delimitada por dos puertas XOR configurables.

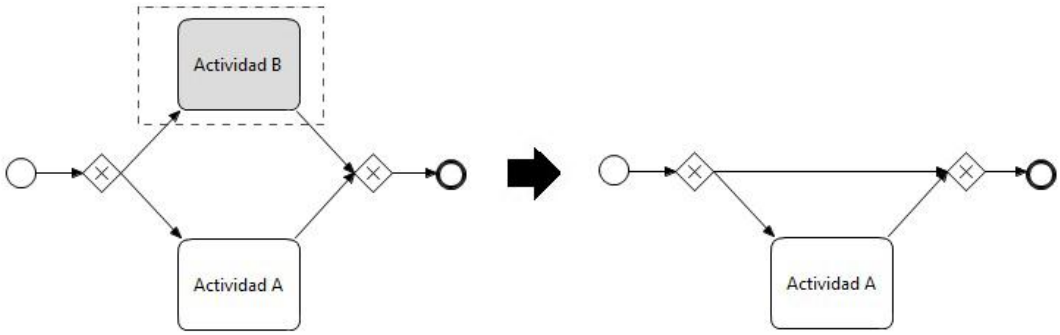


Tabla 14: Implementación de patrón de cambio CP4

Patrón CP8. Insertar Restricción de Configuración
Descripción: Este patrón inserta una Restricción de Configuración.
Condiciones de aplicación: Se asume que este patrón se aplica siempre y solamente entre dos pares de puertas lógicas XOR (que representan dos Regiones Configurables).
Acción: Para implementar este patrón en CEP, se inserta un cuadro de texto editable que queda vinculado (con línea descontinua) a las puertas lógicas XOR que participan en esta restricción. En este cuadro editable se permite escribir la restricción que condiciona la configuración de las Regiones Configurables afectadas. Típicamente, las restricciones tendrán la forma *SEQ1 -> SEQ2* que indica que "Si en la primera puerta lógica XOR, el proceso se configura por la SEQ1 entonces en segunda puerta lógica XOR la configuración será la rama SEQ2". Por este motivo, es necesario nombrar las distintas secuencias posibles dentro de una Región Configurable (ej. SEQ1, SEQ2). En la Figura podemos ver un ejemplo de este patrón.
Instrucciones de aplicación: Para poder aplicar este patrón hay que seleccionar la segunda de las puertas lógicas XOR de la primera Región

Configurable y la primera de las puertas lógicas XOR de la segunda Región Configurable. Si las puertas han sido seleccionadas correctamente se habilita el botón del patrón CP8 en el panel de patrones de CEP. A continuación, basta con apretarlo para que se aplique automáticamente el patrón. Si el botón del patrón esta deshabilitado, este patrón no se puede aplicar ya que los elementos no se han seleccionado correctamente.

Algoritmo: Habilitar Botón de Aplicación del Patrón CP8

```
Algoritmo:
Variables:
Elemento1, Elemento2: Nodo
   SI número de elementos de Área seleccionada = 2 ENTONCES
      Elemento1 <- primer elemento del Área seleccionada
      Elemento2 <- segundo elemento del Área seleccionada
      SI primer elemento del Área seleccionada es XOR y segundo
      elemento del Área seleccionada es XOR ENTONCES
         SI primer elemento del Área seleccionada es "join" y segundo
         elemento del Área seleccionada es "split" ENTONCES
            SI segundo elemento del Área seleccionada sigue después de
            primer elemento del Área seleccionada ENTONCES
               Habilitar el botón
            FINSI
         FINSI
         SI segundo elemento del Área seleccionada es "join" y primer
         elemento del Área seleccionada es "split" ENTONCES
            SI primer elemento del Área seleccionada sigue después de
            segundo elemento del Área seleccionada ENTONCES
               Habilitar el botón
            FINSI
         FINSI
      FINSI
   FINSI
FIN
```

Algoritmo: Implementar Patrón de cambio CP8

```
Algoritmo:
Variables:
Puerta1, Puerta2, Puerta3, Puerta4: Nodo
   Puerta1 <- XOR "join" del Área seleccionada
   Puerta2 <- XOR "split" del Área seleccionada
   Puerta3 <- puerta "split" del elemento Puerta1
```

Puerta4 <- puerta “Join” del elemento Puerta2
Crear nodo Configuración
Crear conexión descontinua de Puerta3 a Configuración
Crear conexión descontinua de Configuración a Puerta1
Crear conexión descontinua de Puerta2 a Configuración
Crear conexión descontinua de Configuración a Puerta4
FIN

En la Figura se puede observar un ejemplo de cómo actúa la aplicación de este patrón. En este ejemplo, el patrón CP8 inserta una Restricción de Configuración entre dos Regiones Configurables delimitadas por puertas XOR.

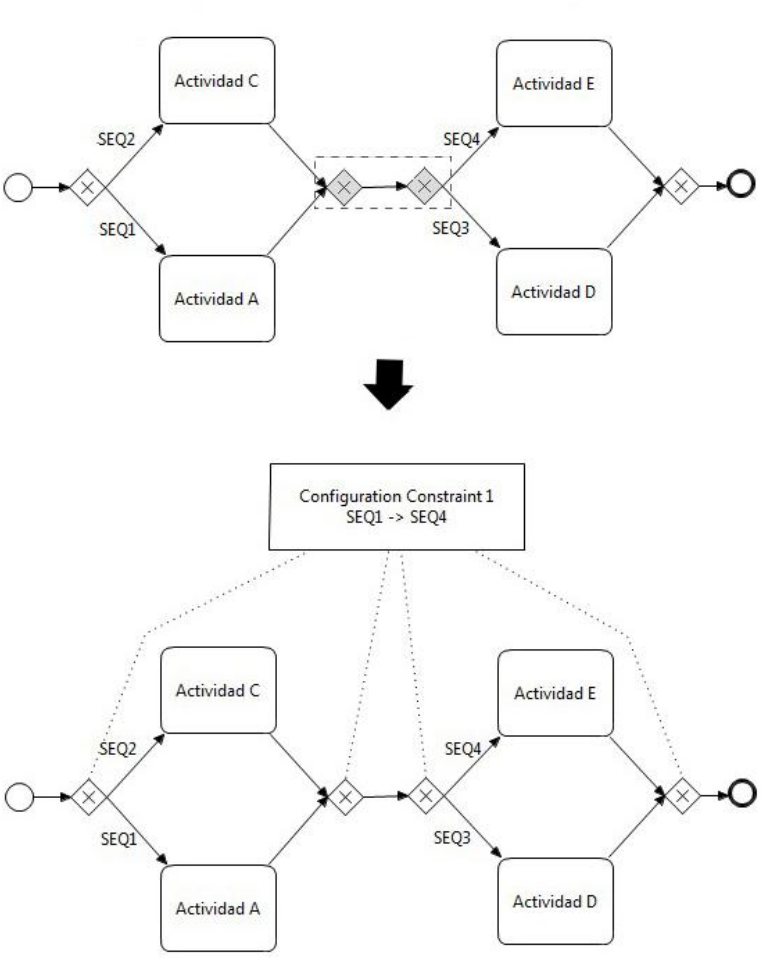


Tabla 15: Implementación de patrón de cambio CP8

Patrón CP9. BORRAR Restricción de Configuración
Descripción: Este patrón borra una Restricción de Configuración.
Condiciones de aplicación: Se asume que este patrón se aplica siempre y solamente a una Restricción de Configuración existente en el modelo de proceso configurable.
Acción: Para implementar este patrón en CEP, se borra la Restricción de Configuración.
Instrucciones de aplicación: Para aplicar este patrón hay que seleccionar una Restricción de Configuración. Si el elemento ha sido seleccionado correctamente, se habilita el botón del patrón CP9 en el panel de patrones de CEP. A continuación, basta con apretarlo para que se aplique automáticamente el patrón. Si el botón del patrón esta deshabilitado, este patrón no se puede aplicar ya que los elementos no se han seleccionado correctamente. **Algoritmo: Habilitar Botón de Aplicación del Patrón CP9** Algoritmo: SI número de elementos de Área seleccionada = 1 ENTONCES SI elemento de Área seleccionada es Restricción de Configuración ENTONCES Habilitar el botón FINSI FINSI FIN **Algoritmo: Implementar Patrón de cambio CP9** Algoritmo: Borrar Restricción de Configuración Borrar conexiones discontinuas FIN En la Figura se puede observar ejemplo de cómo actúa la aplicación de este patrón. En este ejemplo el patrón CP9 borra una Restricción de Configuración.

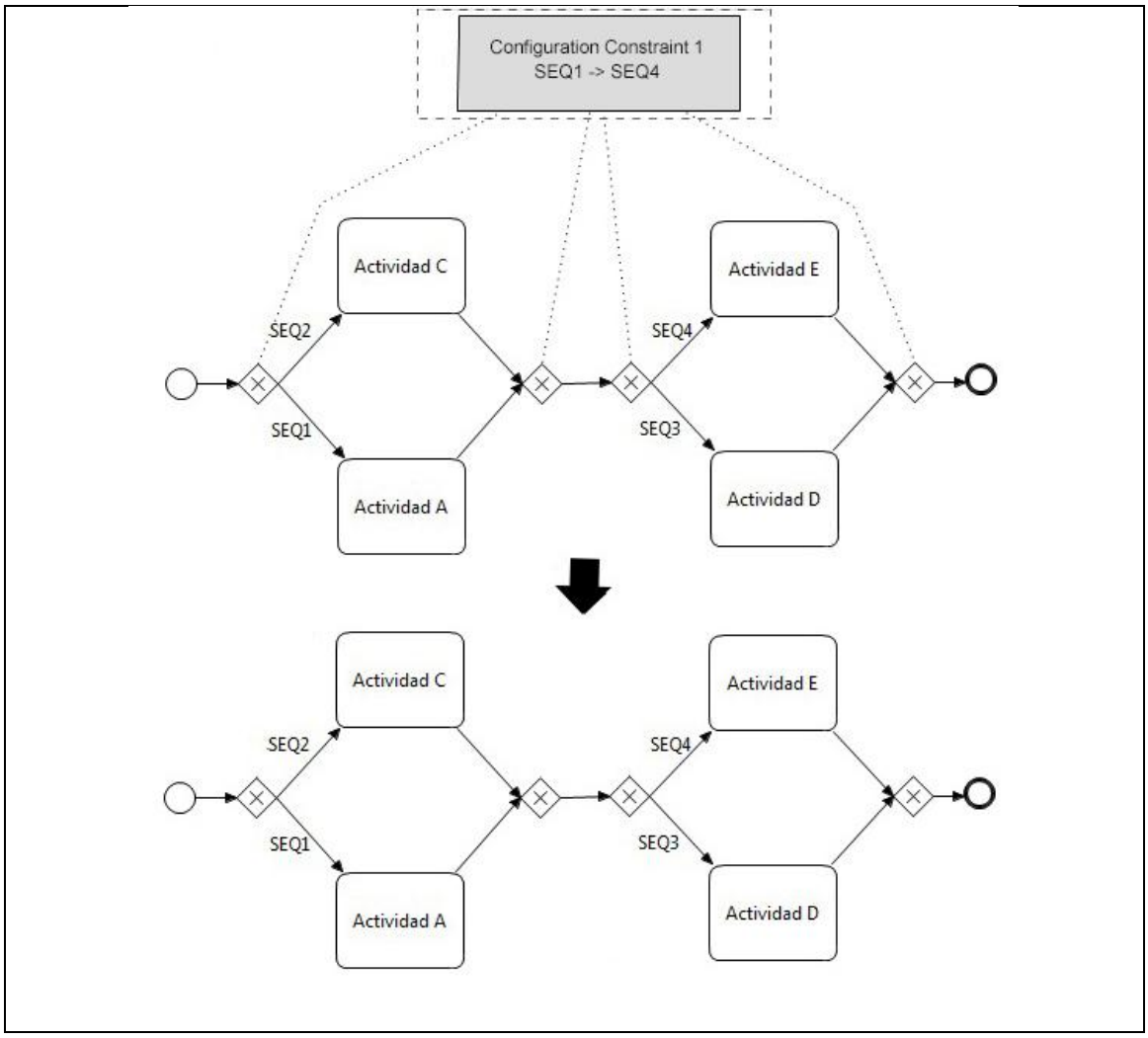


Tabla 16: Implementación de patrón de cambio CP9

CEP, además, proporciona la opción de deshacer la aplicación de un patrón para facilitar la tarea de los modeladores a la hora de gestionar modelos de procesos configurables.

4.2.2 Configuración de una Variante del Proceso

Para mejorar la gestión de las familias de procesos en CEP, fue añadido en CEP una funcionalidad adicional que está ligada con el patrón de cambio CP8 (Insertar Restricción de Configuración) y facilita el uso de este patrón. Se trata de la posibilidad de configurar una posible variante del modelo a través una Restricción de Configuración. En otras palabras se puede ver como se ve el modelo si se cumple una cierta Restricción de

Configuración. Aquí se presenta el algoritmo que implementa esta funcionalidad:

Algoritmo: Habilitar Botón Configuración

```
Algoritmo:
   SI número de elementos de Área seleccionada = 1 ENTONCES
      SI elemento de Área seleccionada es Restricción de Configuración
      ENTONCES
         Habilitar el botón
      FINSI
   FINSI
FIN
```

Algoritmo: Implementar Configuración

```
Algoritmo:
Variables:
Restricción, Puerta, PuertaIF, PuertaTHEN : Nodo
Condicion1, Condicion2: String
ConexionIF, ConexionTHEN : Conexión
Conexiones : Lista de conexiones
NodosBorrar : lista de nodos
   Restricción <- elemento del Área seleccionada
   PARA todos nodos del Modelo HACER
      Hacer copia del Nodo
      Hacer copia del Conexion
   FINHACER
   Condicion1 <- parte derecha del Restricción
   Condicion2 <- parte izquierda del Restricción
   PARA cada conexión en la lista de conexiones que llegan al elemento
   Restricción HACER
      Puerta <- nodo origen de la conexión
      PARA todas conexiones de la lista de conexiones que salen del
      elemento Puerta HACER
         SI nombre de la conexión = Condicion1 ENTONCES
            PuertaIF <- Puerta
            ConexionIF <- conexion
         FINSI
         SI nombre de la conexión = Condicion2 ENTONCES
            PuertaTHEN <- Puerta
            ConexionTHEN <- conexion
         FINSI
```

```
      FINHACER
   FINHACER
   Conexiones <- lista de conexiones que salen del elemento PuertaIF
   PARA todas conexiones de la lista Conexiones HACER
      SI conexión != ConexionIF ENTONCES
         Agregar los nodos de esta rama a la lista NodosBorrar
         Borrar conexion
      FINSI
   FINHACER
   Conexiones <- lista de conexiones que salen del elemento PuertaTHEN
   PARA todas conexiones de la lista Conexiones HACER
      SI conexión != ConexionTHEN ENTONCES
         Agregar los nodos de esta rama a la lista NodosBorrar
         Borrar conexion
      FINSI
   FINHACER
   PARA todos nodos de la lista NodosBorrar HACER
      Borrar nodo
   FINHACER
FINPROGRAMA
```

Para ilustrar esta funcionalidad se utiliza una versión simplificada del proceso de facturación de un billete en un aeropuerto (ver Ejemplo 1).

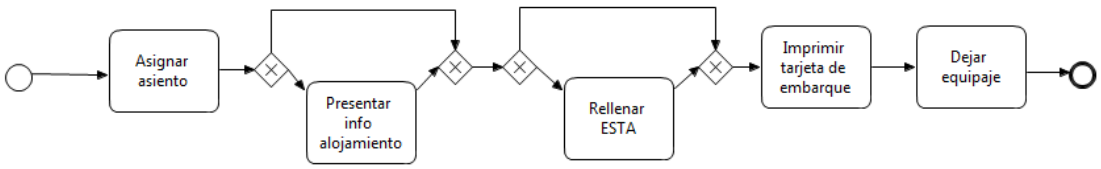


Figura 14: El proceso de facturación de un billete en el aeropuerto

El proceso se inicia cuando se asigna el asiento al pasajero, después se presenta la información sobre alojamiento, si el pasajero viaja a EE.UU. A continuación, si el pasajero ha presentado la información sobre alojamiento, se debe rellenar el formulario ESTA. Después, se imprime la tarjeta de embarque y el pasajero deja su equipaje. En este caso la frase "*si el pasajero ha presentado la información sobre alojamiento, se debe rellenar el formulario ESTA*" representa una Restricción de Configuración entre ambas Alternativas de Configuración (ver Figura 15).

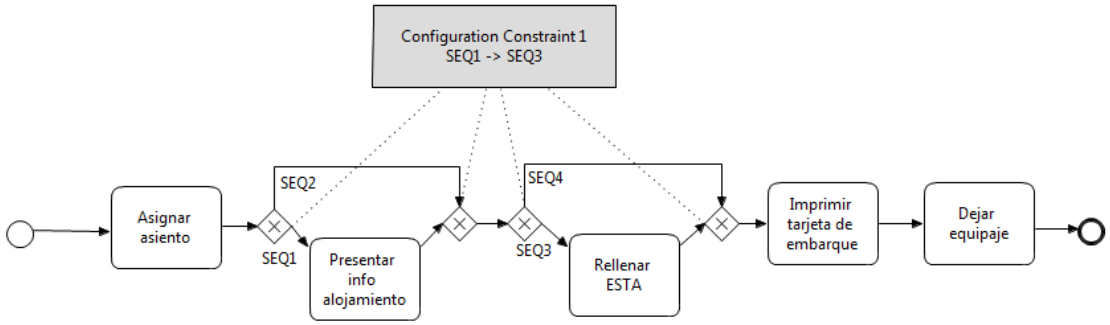


Figura 15: Ejemplo de Restricción de Configuración

Para visualizar una variante del modelo de proceso que cumple la Restricción de Configuración (es decir, la variante del proceso donde el pasajero presenta la información sobre alojamiento y luego rellena el formulario ESTA), fue añadido nueva funcionalidad “Configuración” en el panel de CEP. Para aplicar esta funcionalidad basta con seleccionar la Restricción de Configuración. Esta variante se visualiza debajo del modelo principal (ver Figura 16).

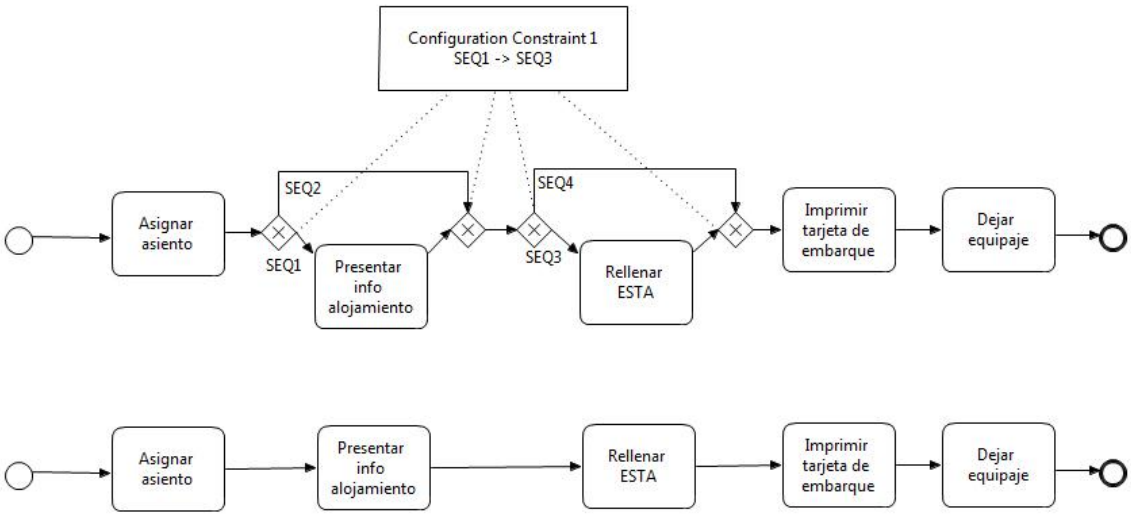


Figura 16: Ejemplo de visualizar una variante del modelo.

4.2.3 Descripción de las Herramientas Implementadas

En este trabajo se propone extender la interfaz del editor de procesos de negocio de CEP para poder implementar los patrones de cambio de familias de proceso descritos en la Sección 3.2.

La estructura interna de CEP se divide en distintos paquetes (o plug-ins) que se representan en la Figura 17:

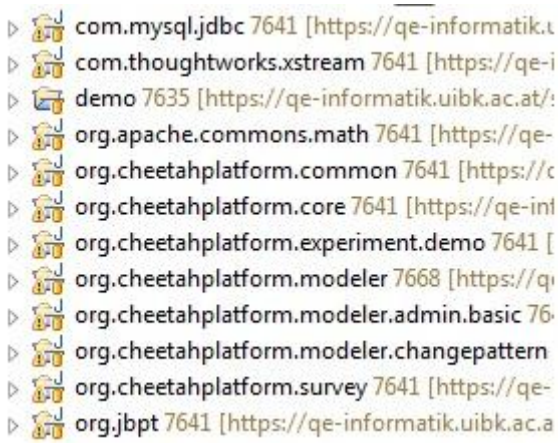


Figura 17: Estructura de CEP

Toda la implementación los patrones de cambio de familias de procesos en CEP ha sido realizada en el paquete *org.cheetahplatform.modeler*. Concretamente, se añadió un nuevo subpaquete (*org.cheetahplatform.modeler.configurablechangepattern)* donde se fueron creando las clases necesarias para dicha implementación.

La clase responsable del panel donde se muestran todos los botones que permiten aplicar los patrones de cambio es *ChangePatternDialogComposite.java*. En esta clase se define qué botones aparecerán en el panel, sus iconos, su nombre, el orden en el panel y el estado por defecto (en este caso, *deshabilitado*). En la Figura 18 se puede observar este panel.

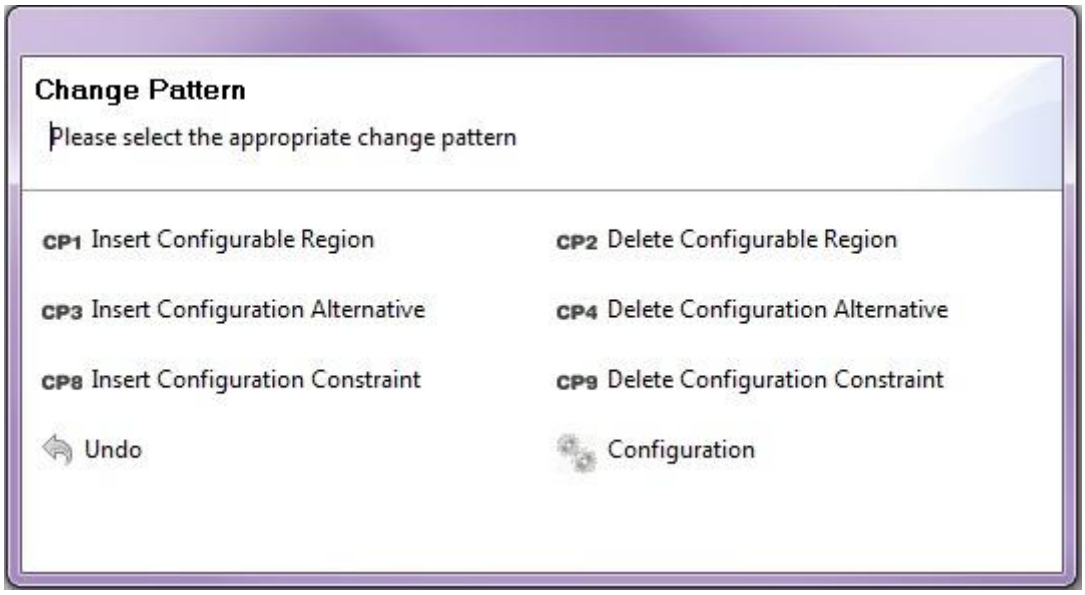


Figura 18: Panel de Patrones de cambio

Para implementar los patrones de cambio, se definen dos clases por patrón. Una clase sirve para *ejecutar los cambios* en el modelo de procesos configurable y disponer de los elementos necesarios para implementar el patrón correspondiente. La otra clase sirve como *gestor de eventos* del patrón. Por ejemplo, el gestor de eventos incluye propiedades tales como habilitar/deshabilitar el botón del patrón correspondiente según los elementos seleccionados (ej., el patrón CP1 se puede aplicar sólo entre dos elementos seguidos secuencialmente, por lo que el botón sólo se habilita si los elementos seleccionados son secuenciales) y llamar a una u otra clase dependiendo del patrón aplicado. Cada clase del gestor de eventos de cada patrón extiende la clase abstracta *AbstractChangePatternAction*.java, sobrescribiendo dos métodos. Además, se sobrescriben dos métodos de dicha clase. El método:

```
protected boolean isChangePatternExecutable(IStructuredSelection selection)
```

Que es responsable de habilitar/deshabilitar el botón correspondiente según los elementos que contiene variable *selection*. El segundo método:

```
protected AbstractChangePattern createChangePattern(IStructuredSelection selection)
```

Que es responsable de llamar a la clase del patrón correspondiente que se va a aplicarse en el área seleccionada en la variable *selection*.

Las clases que ejecutan los patrones de cambio como tal extienden la clase abstracta *AbstractChangePattern.java*. Esta clase abstracta contiene los métodos principales que se usan en las clases heredadas para construir el patrón de cambio correspondiente. Concretamente, estos métodos son:

```
protected Node addAddActivityCommand(Graph graph, String activityName, Point location)
```

Este método agrega al modelo de proceso configurable que se pasa en la variable *Graph* una nueva actividad con el nombre que se pasa en la variable *activityName* y en la posición que se pasa en la variable *location*.

```
protected Node addXorGatewayCommand (Graph graph, Point location)
```

El método agrega al modelo que se pasa en la variable *Graph* (el modelo de proceso configurable) una nueva puerta lógica XOR en la posición que se pasa en la variable *location*.

```
protected Node addAddConditionCommand(Graph graph, String conditionName, Point location)
```

Este método, que se usa sólo para el patrón de cambio CP8 (*Insertar Restricción de Configuración entre unas Regiones Configurables*), agrega al modelo (de proceso configurable) que se pasa en la variable *Graph* un nuevo elemento de Restricción de Configuración con la restricción que se pasa en la variable *conditionName* y en la posición que se pasa en la variable *location*.

```
protected CreateEdgeCommand addAddEdgeCommand(Graph graph, Node source, Node target)
```

Este método se usa en cada patrón de cambio y agrega al modelo (de proceso configurable) que se pasa en la variable *Graph* un nuevo flujo de secuencia desde el nodo *source* (inicio) hasta el nodo *target* (destino).

```
protected CreateEdgeCommand addAddEdgeDashCommand(Graph graph, Node source, Node target)
```

Este método, que sólo se usa en el patrón de cambio CP8 (*Insertar Restricción de Configuración entre unas Regiones Configurables*), agrega al modelo de proceso configurable que se pasa en la variable *Graph* una

nueva línea de conexión descontinua desde el nodo *source* (inicio) hasta el nodo *target* (destino).

En la Tabla 17 se resumen cada una de las clases del paquete *org.cheetahplatform.modeler.configurablechangepattern* que implementan los patrones de cambio para familias de procesos.

Patrón	Clases
CP1	ConfigurableRegion.java ConfigurableRegionAction.java
CP2	DeleteConfigurableRegion.java DeleteConfigurableRegionAction.java
CP3	ConfigationAlternative.java ConfigationAlternativeAction.java
CP4	DeleteConfigurationAlternative.java DeleteConfigurationAlternativeAction.java
CP8	ConfigurationConstraint.java ConfigurationConstraintAction.java
CP9	DeleteConfigurationConstraint.java DeleteConfigurationConstraintAction.java

Tabla 17: Clases que implementan los patrones de cambio para familias de procesos

Además, en la Tabla 18 se resumen los métodos principales que participan en la implementación de los patrones de cambio y las clases a las que pertenecen.

Método	Clase
addAddActivityCommand addXorGatewayCommand addAddConditionCommand addAddEdgeCommand addAddEdgeDashCommand	AbstractChangePattern.java
extractConnectionXOR getConnection isChangePatternExecutable getSESEChecker isInXOR	AbstractChangePatternAction.java

addNodes getFirstNode getIncomingEdge getOutgoingEdge getLastNode isSESEFragment	SESEChecker.java
getAllConnectedEdges getDescriptor getSourceConnections getTargetConnections	Node.java
getDescriptor getSource getTarget setSource setTarget	Edge.java

Tabla 18: Métodos que se llaman para implementar los patrones de cambio.

A continuación, se describen en detalle los métodos de las clases *Node* y *Edge,* ya que son los más utilizados (1) para manejar los elementos del modelo de procesos configurable mostrado en CEP y (2) para implementar los patrones de cambio como tal:

```
public class Node extends GraphElement implements ILocated
```

Esta clase representa los elementos del modelo que son nodos (ej., actividades, puertas, eventos). Sus métodos principales son:

`public List<Edge> getAllConnectedEdges()` - Este método devuelve una lista de flujos de secuencia (entrantes y salientes) del nodo.

`public INodeDescriptor getDescriptor()`- Este método devuelve el *descriptor* del nodo. El descriptor contiene todos los atributos del elemento (id, tipo de elemento, nombre, modelo) y sirve para identificar el propio nodo.

`public List<Edge> getSourceConnections()`- Este método devuelve una lista de flujos de secuencia salientes del nodo.

`public List<Edge> getTargetConnections()`- Este método devuelve una lista de flujos de secuencia entrantes del nodo.

```
public class Edge extends GraphElement
```

Esta clase implementa los flujos de secuencia que conectan los elementos del modelo de proceso configurable. Sus métodos principales son:

`public IEdgeDescriptor getDescriptor()` - Este método devuelve el descriptor del flujo de secuencia. Este descriptor contiene todos los atributos del flujo de secuencia (id, nombre, modelo).

`public Node getSource()` - Este método devuelve el nodo de donde sale este flujo de secuencia (origen).

`public Node getTarget()`- Este método devuelva nodo a donde entra este flujo de secuencia (destino).

`public void setSource(Node source)` - Este método asigna al nodo que se pasa como parámetro *source* este flujo de secuencia como flujo saliente.

`public void setTarget(Node target)` - Este método asigna al nodo que se pasa como parámetro *target* este flujo de secuencia como flujo entrante.

Estas son las principales clases y métodos que se usan en la implementación de los patrones de cambio. Algunos fragmentos de las clases que han sido creadas para implementar los patrones de cambio se pueden ver en el documento *Codigo.pdf*.

5 CASO DE ESTUDIO

En esta Sección se ilustra cómo pueden aplicarse los patrones de cambio para modelar el caso de estudio del proceso de facturación de un billete de avión (ver Ejemplo 1). Esta Sección demuestra la utilidad y eficiencia de la herramienta implementada para aplicar los patrones de cambio para familias de proceso. Concretamente, en la Sección 5.1 se presenta primero como modelar la familia de procesos del proceso de facturación mediante la aplicación de patrones de cambio. A continuación, en la Sección 5.2 se ilustra una configuración de una de las variantes definidas en la familia.

5.1 Modelado de la Familia de Procesos en CEP usando los Patrones de Cambio

Aunque existen diferentes formas de modelar el modelo de proceso configurable que representa la familia de procesos de facturación de billetes (ej. diferente orden de modelado), en este trabajo se presenta la que resulta más intuitiva para el autor del mismo.

El entorno del editor gráfico de CEP se presenta en la Figura 19. En este editor se pueden observar tres zonas: (1) la zona de modelado (ver Figura 19a), (2) la paleta con la lista de patrones de cambio implementados (ver Figura 19b) y (3) la paleta de dibujo con los elementos básicos de BPMN (ej. actividades, eventos) (ver Figura 19c).

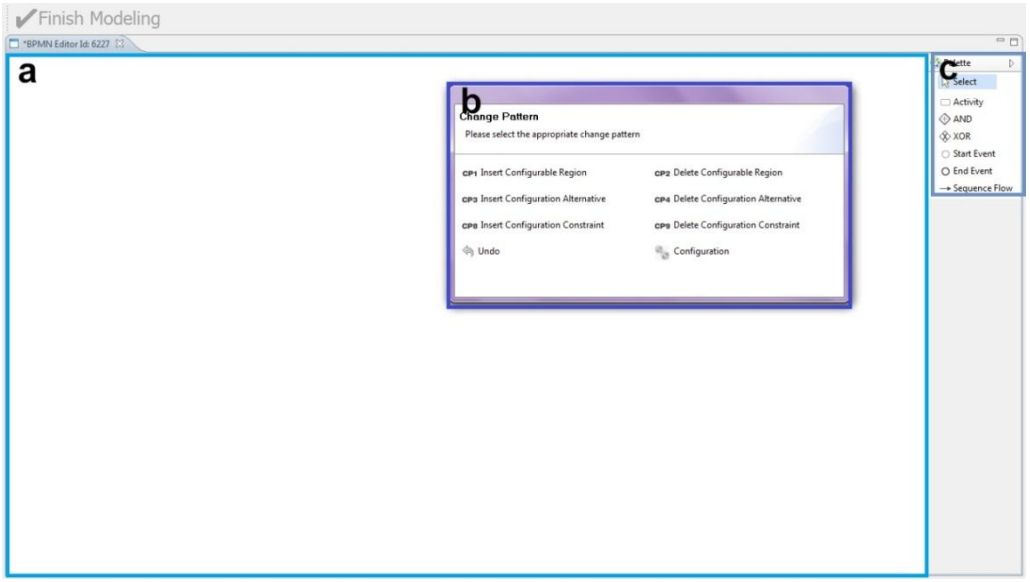


Figura 19 Entorno del editor gráfico de CEP

En primer lugar, se modela los elementos comunes a todas las variantes del proceso de facturación (ver Figura 1 y 2). Estos elementos son las actividades “Identificación de pasajero” e “Imprimir tarjeta de embarque” y los eventos de inicio y fin (ver Figura 20).

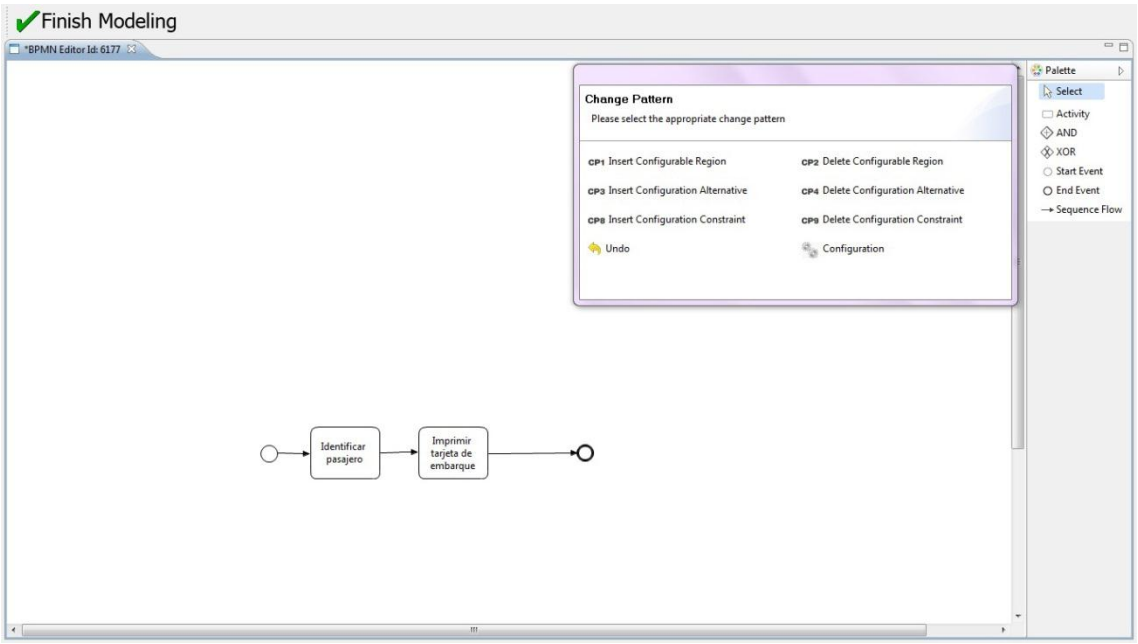


Figura 20: Modelo de proceso configurable con los elementos comunes a todas las variantes

Después de la actividad “Identificar pasajero”, tal y como se describe en el caso de estudio (ver Sección 1) se le asigna asiento al pasajero. Sin embargo, dado que hay varios tipos de pasajeros (ej., menores, personas con discapacidad) existen variaciones en esta actividad. Para reflejar esta variabilidad se debe aplicar el patrón de cambio CP1 (Insertar Región Configurable) que añade una nueva Región Configurable (ver Figura 21).

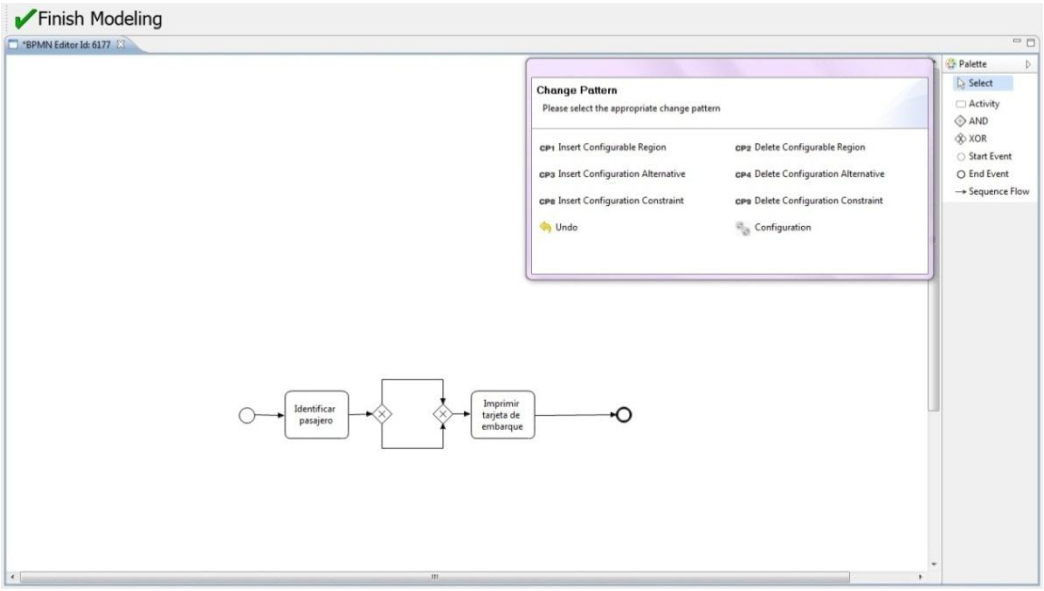


Figura 21: Modelo de proceso configurable después de aplicar el patrón de cambio CP1

Una vez insertada la región, como existen tres variantes de asignación de asiento: (1) asignar asiento para un pasajero ordinario, (2) asignar asiento para una persona con discapacidad y (3) asignar asiento para un menor no acompañado, se aplica tres veces el patrón de cambio CP3 (Insertar Alternativa de Configuración). El resultado de esta aplicación se presenta en la Figura 22.

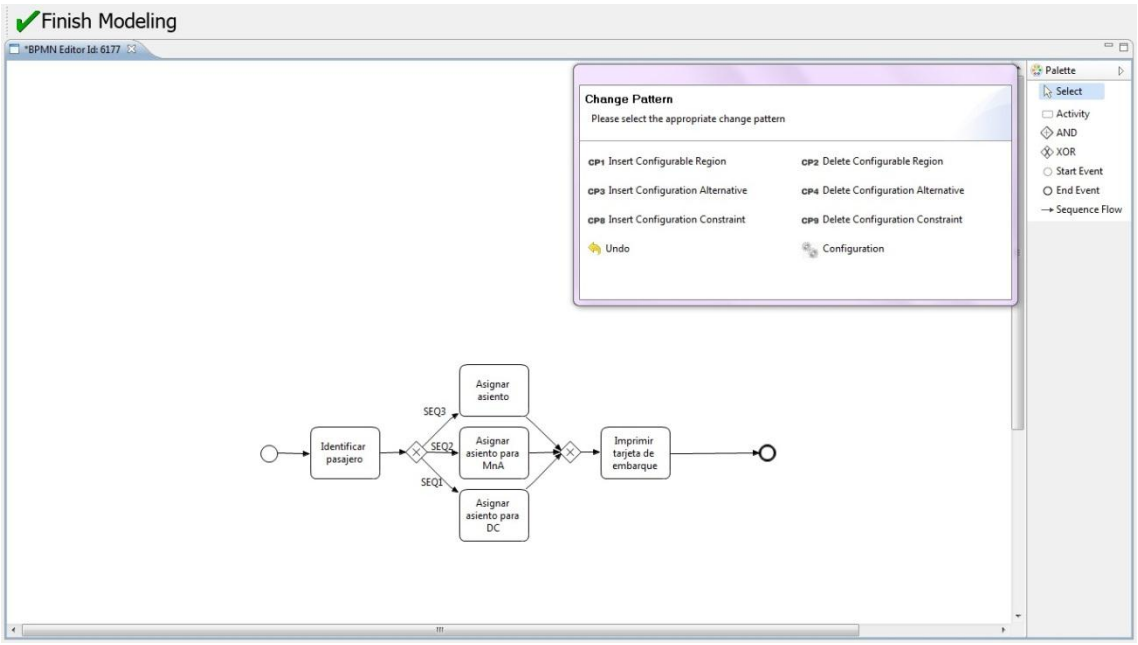


Figura 22: Modelo de proceso configurable después de aplicar el patrón de cambio CP3

En el siguiente paso, se añade la actividad “Rellenar formulario para menor no acompañado”. Como esta actividad es un caso particular (variación), entonces se aplica el patrón CP1 (Insertar Configurable región) y luego el patrón CP3 (Insertar Alternativa de Configuración). Además, aquí se debe indicar que esta actividad sólo ocurre si el pasajero es un menor no acompañado. Para ello, se aplica el patrón CP8 (Insertar Restricción de Configuración) que restringe la ejecución de la actividad “Rellenar formulario para menor no acompañado” si el pasajero es un menor (actividad “Asignar asiento para menor no acompañado”). El resultado puede verse en la Figura 23.

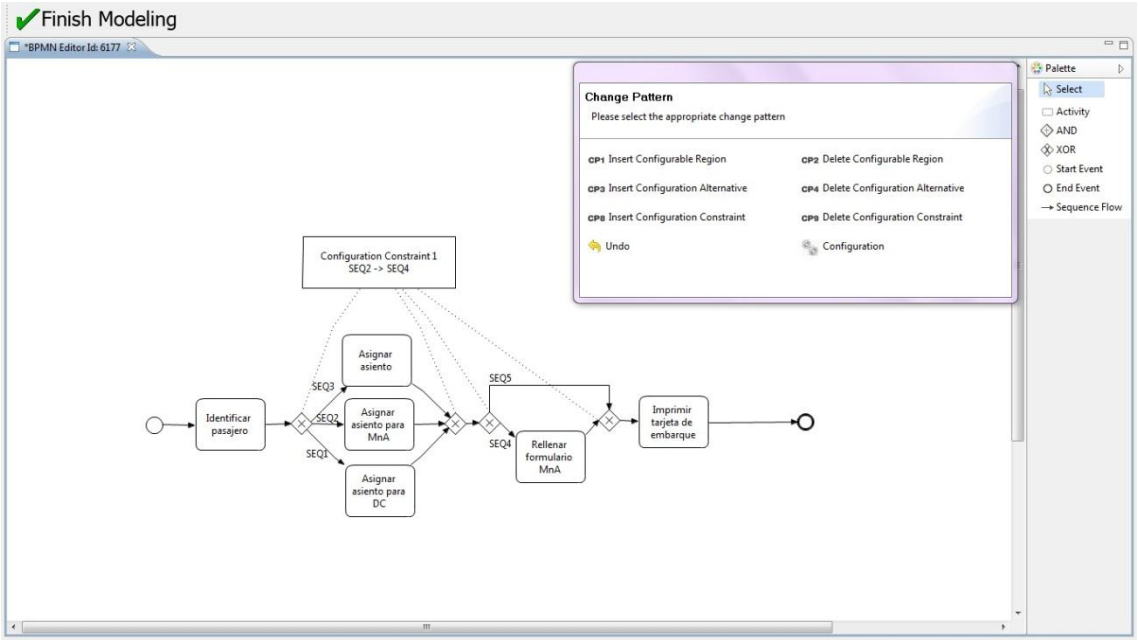


Figura 23: Modelo de proceso configurable después de aplicar el patrón de cambio CP8

Los pasajeros que vuelan a EE.UU. deben presentar información sobre su alojamiento. Para ello, se introduce en el modelo de proceso configurable una región configurable aplicando CP1 y luego una alternativa aplicando CP3 (se introduce actividad “Presentar info Alojamiento”). Además, es necesario rellenar el formulario ESTA. Para esto, se introduce una región configurable aplicando CP1 y luego una alternativa aplicando CP3 (se introduce actividad “Rellenar formulario

ESTA”). Y aplicando el patrón CP8 (Insertar Restricción de Configuración) se indica que si los pasajeros presentan información sobre alojamiento entonces se debe rellenar el formulario ESTA. El resultado puede verse en la Figura 24.

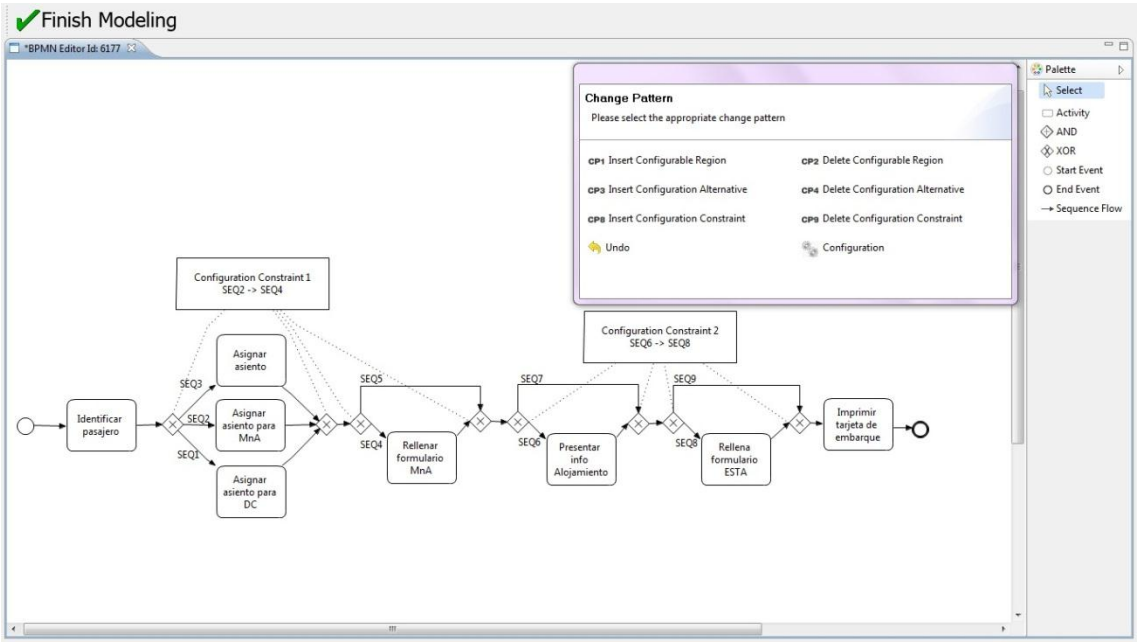


Figura 24: Modelo de proceso configurable después de aplicar el patrón de cambio CP8

Además, para tener en cuenta que puede haber pasajeros con equipaje con sobrepeso (los cuales deben pagar un cargo extra), se debe aplicar el patrón CP1 para introducir *una Región Configurable* y el patrón CP3 para introducir una Alternativa de configuración (actividad “Pagar Suplemento”) (ver Figura 25).

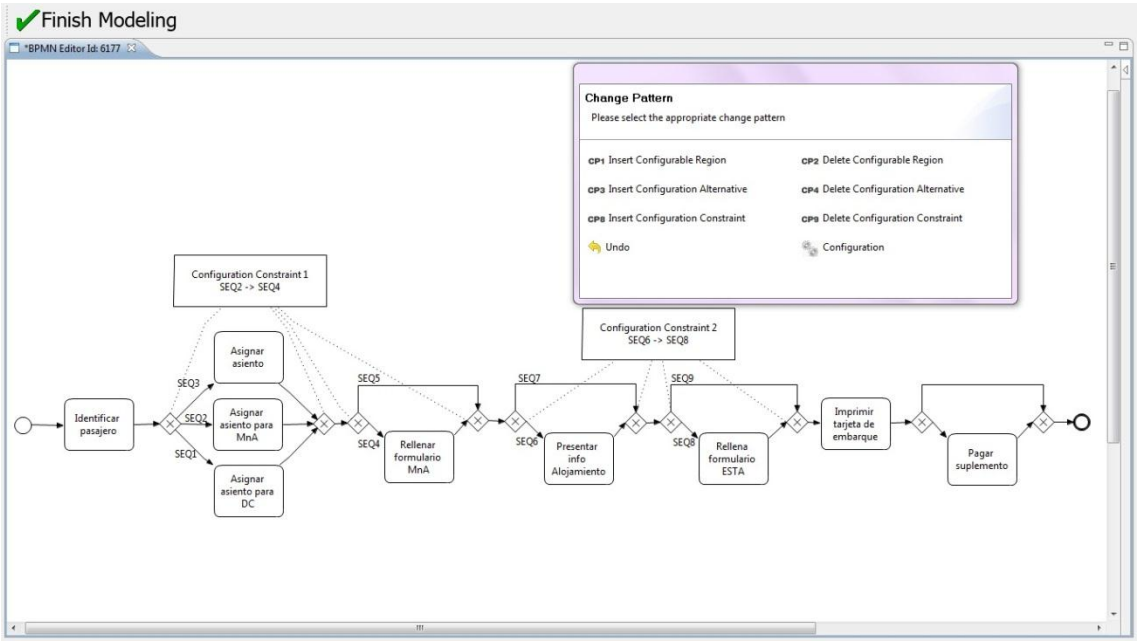


Figura 25: Modelo de proceso configurable después de aplicar los patrones CP1 y CP3

Finalmente, es necesario diseñar la actividad de dejar el equipaje en la cinta correspondiente. Existen dos opciones para dejarlo: (1) dejar equipaje normal y (2) dejar equipaje de gran volumen (ver Figura 26). Para esto, se introduce una región configurable aplicando CP1, una Alternativa de Configuración aplicando CP3 (se introduce actividad “Dejar equipaje normal”) y, finalmente, otra alternativa aplicando CP3 (se introduce actividad “Dejar equipaje de gran volumen”)

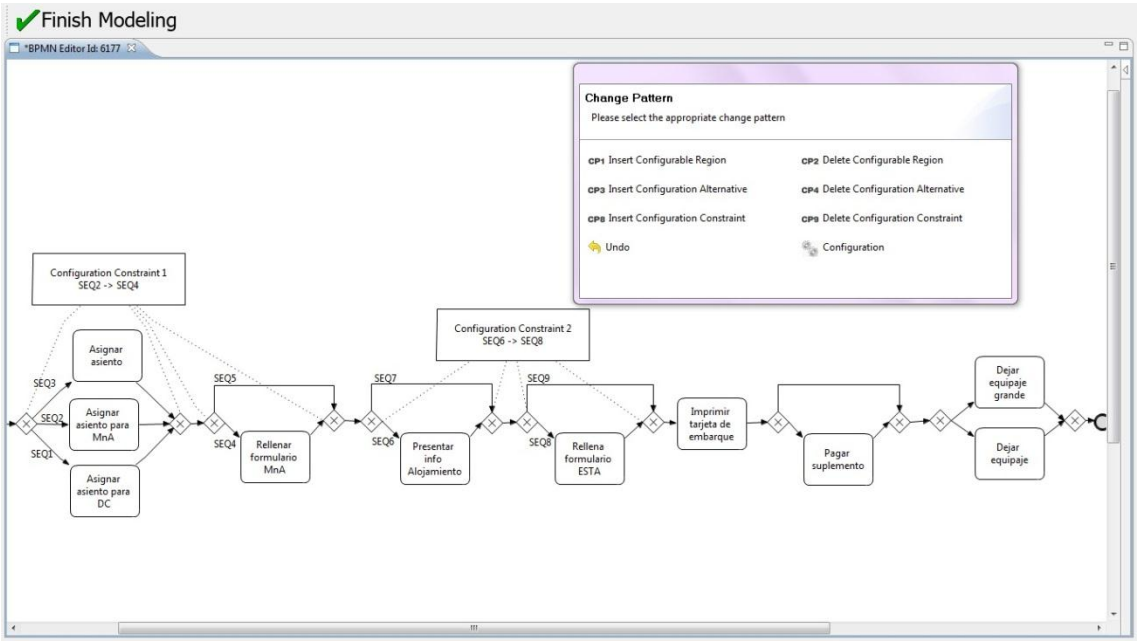


Figura 26: Modelo de proceso configurable después de aplicar los patrones CP1 y CP3

Además, según la definición del caso de estudio, las personas con discapacidad y los menores no acompañados con la actividad “dejar equipaje” tienen que hacer a la vez una actividad más. Para esto, se añaden dos puertas lógicas AND y, entre estas puertas, se añaden la actividad “Dejar equipaje” y una Región Configurable aplicando el patrón CP1. A continuación, aplicando el patrón CP3 se añade una Alternativa de Configuración (se añade la actividad “Imprimir duplicado de tarjeta”) y aplicando otra vez el patrón CP3 se añade otra Alternativa de Configuración (se añade la actividad “Localizar asistente”).

Así, tras la aplicación de los patrones de cambio, el modelo de proceso configurable resultante que representa toda la familia del proceso de facturación de un billete de avión puede observarse en la Figura 27.

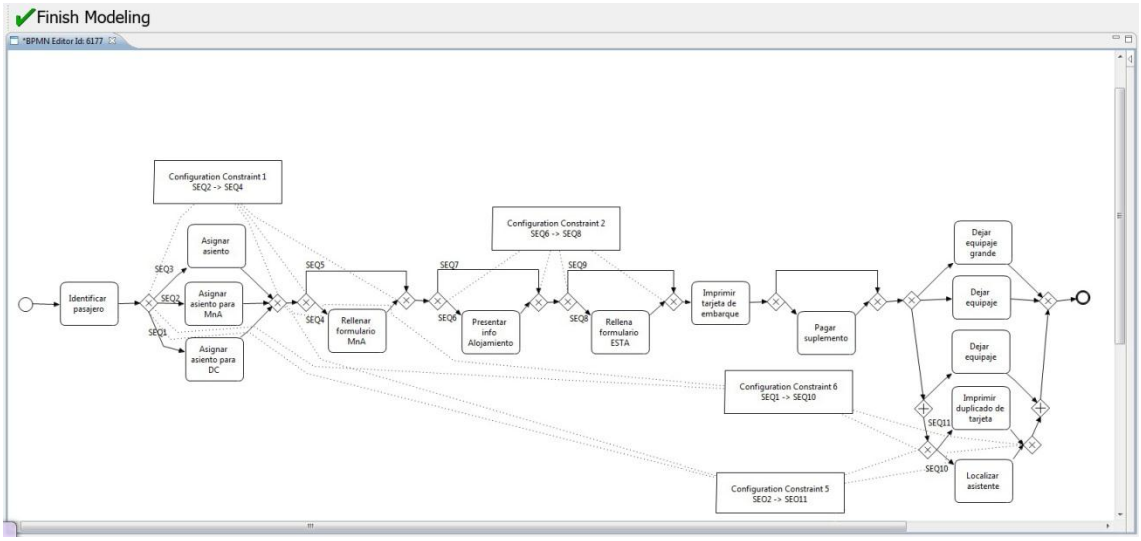


Figura 27: Modelo de proceso configurable resultante después de aplicar los patrones de cambio CP1, CP3 y CP8

En resumen, para modelar la familia de procesos del caso de estudio se han realizado los siguientes pasos:

1. Añadir Evento de inicio,
2. Añadir evento de fin
3. Añadir Actividad común “Identificar pasajero”
4. Añadir Actividad común “Imprimir tarjeta de embarque”
5. Aplicar patrón CP1
6. Aplicar patrón CP3 (“Asignar Asiento”).
7. Aplicar patrón CP3 (“Asignar Asiento para MnA”).
8. Aplicar patrón CP3 (“Asignar Asiento para DC”).
9. Aplicar patrón CP1
10. Aplicar patrón CP3 (“Rellenar formulario MnA”).
11. Aplicar patrón CP8 (Restricción de configuración 1).
12. Aplicar patrón CP1
13. Aplicar patrón CP3 (“Presentar info Alojamiento”).
14. Aplicar patrón CP1
15. Aplicar patrón CP3 (“Rellenar formulario ESTA”).
16. Aplicar patrón CP8 (Restricción de configuración 2).
17. Aplicar patrón CP1
18. Aplicar patrón CP3 (“Pagar Suplemento”).
19. Aplicar patrón CP1

20. Aplicar patrón CP3 (“Dejar Equipaje”).
21. Aplicar patrón CP3 (“Dejar Equipaje grande”).
22. Añadir puerta AND.
23. Añadir puerta AND.
24. Añadir Actividad “Dejar equipaje”.
25. Aplicar patrón CP1
26. Aplicar patrón CP3 (“Imprimir duplicado de tarjeta”).
27. Aplicar patrón CP3 (“Localizar asistente”).
28. Aplicar patrón CP8 (Restricción de configuración 3).
29. Aplicar patrón CP8 (Restricción de configuración 4).

Si no se hubieran utilizado estos patrones y se hubieran usado los elementos básicos del lenguaje, el número de pasos necesitados hubiera sido, aproximadamente, 90. Esto es debido a que, por cada elemento del modelo, se hubiera necesitado utilizar una primitiva de BPMN para modelarlo. Por ejemplo, se hubiera necesitado un paso para incluir el evento de inicio, otro paso para incluir la Actividad “Identificar pasajero”, otro para el flujo de secuencia entre ambos, otro para poner una puerta lógica XOR, etc.

5.2 Configuración de una Variante del Proceso

Como en el modelo de proceso configurable final se han definido varias Restricciones de Configuración, se puede obtener una posible variante de proceso que cumple una Restricción de Configuración concreta con la funcionalidad de Configuración implementada. Por ejemplo, en la Figura 28 se puede ver la configuración de la variante obtenida de la restricción de configuración 1 (SEQ2 -> SEQ4).

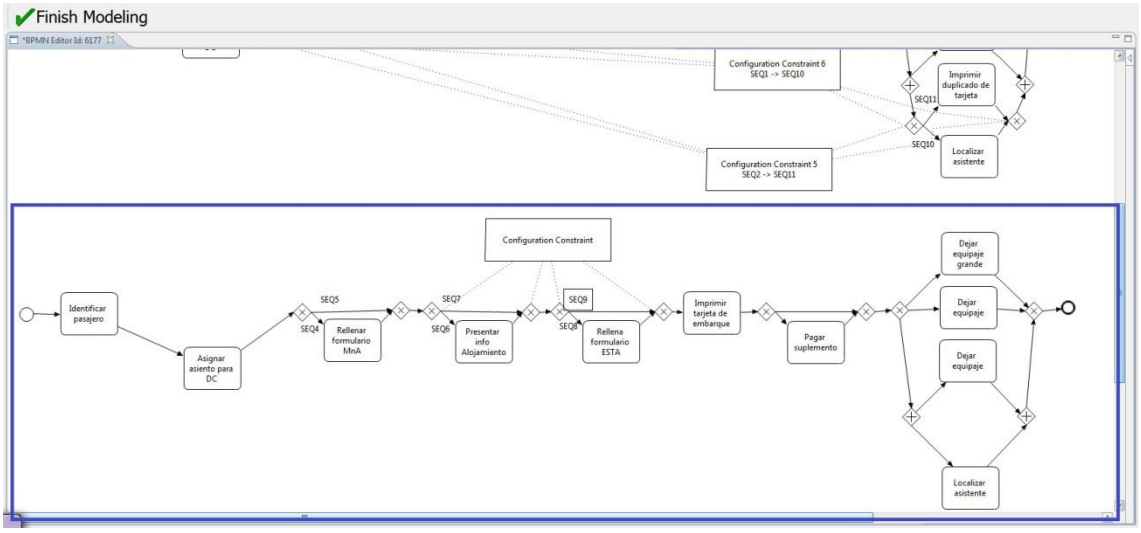


Figura 28: Configuración de una posible variante de proceso

6 CONCLUSIONES

El objetivo de este trabajo ha sido extender el editor gráfico de CEP para implementar los patrones de cambio para familias de procesos. Aunque no se han implementado todos los patrones de cambio definidos, este trabajo contribuye significativamente al problema de la variabilidad en las familias de proceso. Por ejemplo, la implementación realizada permite aplicar de manera más rápida y eficaz los patrones (si se compara con su aplicación sobre modelos en papel). Además, la implementación realizada permite estudiar el impacto de los patrones de cambio en el modelado de las familias de procesos, así como en el cambio, ya sea en diseño o en tiempo de ejecución. Por ejemplo, esta implementación permite llevar a cabo experimentos para medir el esfuerzo necesario para manejar la variabilidad en las familias de procesos.

Por otro lado, con el caso de estudio presentado, se demuestra que el uso de los patrones de cambio resulta sencillo y que los patrones permiten a los diseñadores de los procesos de negocio gestionar las familias de proceso a lo largo del tiempo, reduciendo el esfuerzo necesario para modelar una familia de procesos. Finalmente, el modelado de los procesos variables resulta más cómodo, menos costoso y más fácil con patrones de cambio.

El desarrollo de este trabajo final de master ha sido bastante complejo. Aunque, se disponían de conocimientos de modelado de procesos de negocio y de notación BPMN, vistos en algunas asignaturas de master (**IDP**, **MSW**, **MSI**), apenas se poseían conocimientos sobre variabilidad en los procesos de negocio ni en los patrones de cambio. Durante el proceso de desarrollo de este trabajo, gracias al esfuerzo realizado, se han obtenido conocimientos sobre variabilidad en las familias proceso de negocio y en los métodos de gestionarla mediante patrones de cambio. Por tanto, el alumno está muy satisfecho con la labor realizada en el trabajo, y sobre todo, por los nuevos conocimientos adquiridos como la variabilidad en los procesos de negocio, patrones de cambio y la experiencia de la ampliación de una herramienta ya implementada.

6.1 Trabajos Futuros

Ya que el lenguaje de modelado soportado por CEP (BPMN) no da soporte a la variabilidad en los procesos de negocio como tal (ej. no se permite definir condiciones de contexto), no se ha implementado todo el conjunto de patrones de cambio para familias de procesos definidos en [4]. Por esto, un posible trabajo futuro es implementar los patrones de cambio restantes:

- CP5: Insertar Condición del Contexto de una Alternativa de Configuración.
- CP6: Borrar Condición del Contexto de una Alternativa de Configuración.
- CP7: Modificar Condición del Contexto de una Alternativa de Configuración.

Sin embargo, para poder implementar estos patrones, el lenguaje BPMN debería ser extendido para dar soporte a las condiciones del contexto. En concreto, sería necesario añadir la posibilidad de introducir una condición en las puertas lógicas XOR. Una posible solución es cambiar el funcionamiento de las anotaciones (elemento BPMN) y que éstas puedan afectar al flujo de secuencia (ahora no afectan).

Otro posible trabajo futuro seria ampliar la funcionalidad de "Configuración" actualmente implementada. Sería muy interesante dar la posibilidad al usuario de elegir *varias* Restricciones de Configuración para visualizar una posible variante del proceso entero. Actualmente, sólo se da soporte a elegir única Restricción de Configuración.

Otra línea de trabajo futuro podría ser la realización de experimentos que permitieran evaluar de manera cualitativa el uso de los patrones de cambio. Por ejemplo, al igual que se demostró en [5], podría experimentarse con el modelado de familias de procesos sin y con patrones de cambio. Este tipo de experimentos permitiría conocer los retos que supone el uso de patrones de cambio en familias de procesos así como las ventajas que introduce en la creación y mantenimiento de dichos procesos.

Otra posible línea de trabajo futuro está relacionada con mejoras del editor gráfico de CEP. Actualmente, CEP no da soporte a las tareas no atómicas (subprocesos). Sería interesante incorporar en el editor gráfico de CEP la posibilidad de gestionar los subprocesos (ej. minimizar los fragmentos del modelo a un subproceso). También sería de gran utilidad añadir la posibilidad de copiar los fragmentos del modelo y poder reutilizarlos. Esto permitiría a los usuarios utilizar CEP de forma más fácil y sencilla.

REFERENCIAS

1. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: Challenges, scenarios, algorithms. Data Knowledge & Engineering 70(5), pp. 409–434 (2011).
2. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach. J. of Software Maintenance 22(6-7), pp 519–546 (2010).
3. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modeling language. Information Systems 32(1), pp 1–23 (2007).
4. Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: Enhancing Modeling and Change Support for Process Families through Change Patterns. BMMDS/EMMSAD 2013: pp 246-260.
5. Weber B., Pinggera J, Torres V., Reichert M.: Change Patterns in Use: A Critical Evaluation. BMMDS/EMMSAD 2013, pp. 261-276.
6. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - Enhancing flexibility in process-aware information systems. Data Knowledge & Engineering 66(3), pp. 438-466 (2008).
7. Wil, M. P. van der Aalst: Workflow Patterns. Encyclopedia of Database Systems 2009, pp. 3557-3558.
8. Reichert, M., Weber, B.: Enabling flexibility in process-aware information systems: challenges, methods, technologies. Springer (2012).
9. Gosling, J., Joy, B., Steele, G., Bracha, G. and Buckley, A.: “The Java™ Language Specification Java SE 7 Edition” 2012.
10. Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT Support for Release Management Processes in the Automotive Industry. In: Proc. of the 4th Int. Conf. on Business Process Management (BPM’06) pp. 368–377 (2006).
11. Eclipse Foundation. Eclipse. [Online]. http://eclipse.org
12. Eclipse Foundation. Eclipse Platform Overview. [Online]. http://www.eclipse.org/eclipse/presentation/eclipse-slides.pdf
13. Gschwind, T., Koehler, J., Wong, J.: Applying patterns during business process modeling. In: Proc BPM’08, pp. 4–19 (2008).

14. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Computer Science R&D 23, pp. 81–97 (2009).
15. Grambow, G., Oberhauser, R., Reichert, M.: Contextual injection of quality measures into software engineering processes. International Journal of Advanced Software Engineering 4, pp. 76-99 (2011).
16. Weber, B., Reichert, M., Reijers, H.A., Mendling, J.: Refactoring large process model repositories. Computers in Industry 62(5), pp. 467–486 (2011).
17. Aghakasiri, Z., Mirian-Hosseinabadi, S.H.: Workflow change patterns: Opportunities for extension and reuse. In Proc. SERA'09, pp. 265-275 (2009).
18. Küster, J., Gerth, C., Forster, A., Engels, G.: Detecting and resolving process model differences in the absence of a change log. In Proc. BPM'08, pp. 244–260 (2008).
19. IBM: IBM WebSphere Business Modeller, Version 6.1. (2007)
20. Bizagi Process Modeler. [Online] http://help.bizagi.com/processmodeler/es/
21. Mayer, R. J., Painter, M. K., & DeWitte, P. (1992). IDEF family of methods for concurrent engineering and business re-engineering applications. College Station, TX: Knowledge Based Systems, Inc.
22. Dumas, M., & ter Hofstede, A. H. M. Uml activity diagrams as a workflow specification language. In UML'01: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, pp. 76–90 (2001).
23. Hofreiter, B., Huemer, C., & Klas, W. ebxml: Status, research issues, and obstacles. In Proc. Of 12th Int. Workshop on Research Issues on Data Engineering (RIDE02), pp. 7–16, 2002.
24. Business Process Modeling Notation (BPMN) Specification. OMG Final Adopted Specification. formal/2011-01-03.
25. Pinggera J., Zugal S., Weber B.: Investigating the Process of Process Modeling with Cheetah Experimental Platform. In: Proc. ER-POIS '10, pp. 13–18, 2010.
26. Garimella K., Lees M., Williams B.: Introducción a BPM para Dummies. 2008.

27. Object Management Group. [Online] http://omg.org
28. Weske, M.: Business Process Management: Concepts, Languages, Architecture.
29. Weber B., Pinggera J, Torres V., Reichert M.: Change Patterns in Use: A Critical Evaluation. BMMDS/EMMSAD 2013: pp. 261-276.
30. http://www.redhat.com/
31. http://www.oracle.com
32. http://www.hp.com
33. Rodríguez, E.: "Implementación de BPM como herramienta de integración y administración de una organización", La Universidad Católica de Loja, Ecuador, 2011.
34. http://www.ibm.com
35. www.adobe.com
36. www.sap.com
37. www.oracle.com/us/sun/index.html
38. Gottschalk, F., van der Aalst, W., Jansen-Vullers, M. H.: Configurable process models - A foundational approach. Reference modeling, Physica Verlag HD, pp. 59–77, (2007).
39. Schnieders, A., Puhlmann F.: Variability modeling and product derivation in e-business process families. Technologies for Business Information Systems, pp. 63–74, (2007).