UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Workflows flexibles para procesos de desarrollo de software

Autor: Carlos del Fresno Canales

Director: Dr. Patricio Letelier Torres

Julio de 2010

Tesis presentada para cumplir con los requisitos finales para la obtención del título de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información, de la Universidad Politécnica de Valencia, 2010.

Agradecimientos

De forma muy especial a mis padres y a mi hermano que me han apoyado siempre y han creído en mí aún cuando yo no lo hacía.

A Patricio Letelier, por la gran oportunidad que me ha dado, por todo lo que me ha enseñado, por su dedicación y ayuda y por la confianza que siempre ha depositado en mí. Gracias por esta gran oportunidad.

A mis amigos de toda la vida, Virginia, Joan, Ximo y Vicen por haberme ayudado más de lo que creen ya que son una parte muy importante de mí.

Por último, dar las gracias a todos mis compañeros de laboratorio, los cuales me han ayudado a mejorar tanto personal como profesionalmente.

Y en último lugar y más importante, a mi pareja, Elena, por soportarme en los malos momentos y darme los ánimos necesarios para hacerme continuar cuando más los necesitaba. Gracias. Sin ti no podría haberlo logrado.

Contenido

Agradecimientos 3

Capítulo 1. Introducción 9

1.1 Motivación 9

1.2 Organización de la memoria 10

Capítulo 2. Introducción a los Flujos de Trabajo 11

2.1 Introducción 11

2.2 Sistemas de Gestión de Flujos de Trabajo. 13

2.2.1 Taxonomía 14

2.2.2 Limitaciones 19

2.2.3 Estandarización 20

2.2.3.1 Modelo de Referencia de la WfMC 22

2.3 Conclusiones 24

Capítulo 3. Modelos de Flujos de Trabajo 27

3.1 Representación de Flujos de Trabajo 27

3.1.1 Características de los lenguajes de representación de Flujos de Trabajo 27

3.2 Interpretación de Flujos de Trabajo 32

3.2.1 Representación e Interpretación 34

3.2.2 Problemas de la interpretación de Flujos de Trabajo 36

3.3 Selección de un Sistema de gestión de Flujos de Trabajo 38

3.3.1 TUNE-UP 38

3.3.2 Modelos Orientados a la Representación 39

3.3.3 Modelos orientados a la interpretación de Flujos de Trabajo 41

Capítulo 4. Business Process Modeling Notation 45

4.1 ¿Qué es BPMN? 45

4.2 Fundamentos de BPMN 45

4.3 Objetos de flujo 46

4.3.1 Objetos conectores 47
4.3.2 Swimlanes (canales) 49
4.3.3 Artefactos 51
4.4 Uso General de BPMN 53
4.4.1 Procesos B2B colaborativos 53
4.4.2 Procesos de negocio internos 54
4.5 Diferentes niveles de precisión 54
4.6 Valor de modelar en BPMN 56
4.7 Mapear un diagrama BPMN a BPEL4WS 56
4.8 El futuro de BPMN 57
4.9 Notación BPMN 58
4.9.1 Compuertas 58
4.9.2 Actividades 60
4.9.3 Datos 62
4.9.4 Transacciones 63
4.9.5 Documentación 64
4.9.6 Swimlanes 64
4.9.7 Tipología de Eventos 65
Capítulo 5. Tecnologías de Workflows Actuales 67
5.1 Windows Workflow Foundation 67
5.1.1 Introducción 67
5.1.2 Tipos de Flujos de Trabajo soportados 67
5.1.3 Ejemplo de Secuenciación 68
5.1.4 Ejemplo de puesta en marcha 70
5.2 jBPM 76
5.2.1 Introducción 76
5.2.2 Características 76

5.2.3 Diseñador gráfico de proceso jBPM 77
5.2.4 Ejemplo de puesta en marcha 78
Capítulo 6. La metodología TUNE-UP 83
6.1 Introducción 83
6.2 TUNE-UP Process Tool 87
6.2.1 Planificador personal 87
6.2.2 Gestor de Unidades de Trabajo (WUM) 87
6.2.3 Version Content & Tracking (VCT) 89
6.3 Mecanismos de colaboración 90
6.3.2 Utilización de los saltos 91
6.3.3 Estado de la Actividad 92
6.3.4 Cambio de Workflow 92
6.3.5 Cambio de Agente en un Actividad 92
6.3.6 Trabajo en Paralelo 92
Capítulo 7. Workflows en TUNE-UP 93
7.1 Descripción del sistema 93
7.2 Definición de workflows 94
7.3 Asignación de Workflows a productos 97
7.4 Asignación de Agentes por defecto 98
7.5 Asignación de Agentes por unidad de trabajo 99
7.6 Visualización en el PEP de la actividad-estado de una unidad de trabajo 100
7.7 Visualización en VCT de la actividad –estado de una unidad de trabajo 101
7.8 Registro de tiempos 101
7.9 Dashboard 102
7.10 Situaciones excepcionales tratadas por código 103
7.11 Frameworks Utilizados 106
7.11.1 CodeSmith 106

7.11.2 .NetTiers 106
7.11.3 Manual de generación de código mediante .NetTiers y CodeSmith 107
Capítulo 8. Workflows en Acción 113
8.1 Saltos 113
8.2 Cambio de Agente 115
8.3 Trabajo en paralelo 116
8.4 Añadir nuevas actividades 118
Capítulo 9. Conclusiones y trabajo futuro 121
Referencias 127
Anexo 1. Patrones Workflow 133
A.1. Patrones básicos de control de flujo 133
A.2. Patrones de Ramificación Avanzada y Sincronización 136
A.3. Patrones Estructurales 139
A.4. Patrones que involucran Múltiples instancias 139
A.5. Patrones Basados en Estados 140
A.6. Patrones de Cancelación 142
Anexo 2. Ejemplo de workflows implantados 143

Capítulo 1. Introducción

1.1 Motivación

A pesar de los avances en tecnología de workflows y de su aceptación en sistemas colaborativos e integración en aplicaciones, en el ámbito del desarrollo de software los workflows no han sido utilizados de forma significativa. Esto a pesar de que en desarrollo de software debe orquestarse el trabajo de todos los participantes del equipo. En nuestra opinión, el principal impedimento ha sido la rigidez de los workflows.

Existen actualmente multitud de sistemas de gestión de workflows comerciales, pero todos ellos carecen de una característica fundamental para aplicarse en procesos de desarrollo de software: flexibilidad. No suelen permitir acciones tales como hacer saltos no predefinidos de una actividad a otra, permitir cambios de agentes asignados, trabajar en paralelo, etc.

La necesidad de flexibilidad en el ámbito del desarrollo de software es imprescindible debido a que se trata de un contexto muy dinámico; cambian los requisitos, el re-trabajo es inevitable, las prioridades se modifican constantemente, etc. Todo esto se hace aún más evidente al trabajar con metodologías ágiles las cuales incluso fomentan dicho dinamismo para asegurar que el sistema se adapte a las necesidades cambiantes del cliente.

El trabajo de tesis que se va a presentar ha sido desarrollado en una PYME, en el marco de un convenio universidad-empresa. Esta PYME es una empresa de desarrollo de software que desarrolla un ERP dirigido al al sector socio-sanitario y cuenta con más de 40 empleados y más de 850 clientes.

El objetivo global de esta tesis es crear un motor de workflows flexibles para el proceso de desarrollo de software que esté centrado en el proceso, esté orientado a las personas para que trabajen de forma coordinada y sobre todo, sea colaborativo para integrarlo en la metodología TUNE-UP y en la herramienta de apoyo al proceso de desarrollo de softwarey que da soporte a dicha metodología. Como objetivos secundarios sería la de integrar el motor desarrollado con otras funcionalidades tales como la planificación y seguimiento de procesos, la asignación y balanceo de carga de agentes entre otros como validar la efectividad de los workflows flexibles.

1.2 Organización de la memoria

A continuación se presenta la estructura de la tesis de máster:

El capítulo 1 incluye una introducción de la motivación del trabajo, explicando las necesidades a cubrir con el módulo desarrollado.

En el capítulo 2 se realiza una introducción sobre los workflows y los sistemas que los gestionan, con su taxonomía, limitaciones y estándares que existen.

En el capítulo 3 se va a exponer el problema de la representación e interpretación de los flujos de trabajo. Finalmente se analizarán modelos de workflows existentes centrándose en sus ventajas e inconvenientes para su aplicación y sobre todo comparándolo con la metodología TUNE-UP.

En el capítulo 4 se hace una introducción a la notación BPMN comentando tanto las características como los diferentes conectores que existen.

En el capítulo 5 se hace una introducción a las tecnologías que existen actualmente en el mercado, haciendo especial mención a Windows Workflow Foundation y a jBPM (la alternativa Java).

El capítulo 6 explica los aspectos básicos de la metodología TUNE-UP y también ofreciendo una visión global de su herramienta, el TUNE-UP Software Process.

El capítulo 7 se centra en mostrar la arquitectura del motor de workflow, con el esquema de la base de datos como mostrando las situaciones que se han tenido que hacer mediante programación. También se hace referencia a los frameworks y generadores de código que se han utilizado para parsear la base de datos.

En el capítulo 8 se muestran unos manuales de utilización del motor de workflows, en donde se explotan las características que hacen que el motor implementado adquiera esa flexibilidad deseada.

El capítulo 9 concluye el trabajo mediante las conclusiones obtenidas en el proceso y el trabajo futuro surgido a raíz de esta tesis de máster.

Capítulo 2. Introducción a los Flujos de Trabajo

2.1 Introducción

El contenido de este apartado ha sido extraído de la tesis doctoral de M.C. Penadés [58].

En los años 70, 80 y parte de los 90, el principal uso de los ordenadores en las organizaciones era la automatización de las actividades individuales. La situación actual es distinta, en los últimos años hay un interés cada vez más creciente por las organizaciones vistas como un todo. Existen métodos y notaciones de análisis y diseño consolidados, como UML [4], herramientas de desarrollo cada vez más potentes (RAD, generadores de código, etc.) y modelos de ciclo de vida basados en el uso de las mismas. Todo ello ha motivado que se esté produciendo un cambio de orientación hacia los procesos organizacionales o procesos de negocio. El interés de las organizaciones no está limitado únicamente al desarrollo de software que automatice ciertas actividades individuales, sino que su objetivo final es la automatización de todo el proceso de negocio, puesto que de ello depende gran parte su competitividad. Surgen, por lo tanto, nuevas necesidades de capturar, modelar, ejecutar y monitorizar los procesos de negocio, vistos como un conjunto de procedimientos o actividades enlazadas, cuya realización permite alcanzar un cierto objetivo o meta en el contexto de una organización [14, 16]. En este nuevo escenario, los flujos de trabajo y la tecnología asociada ofrecen un marco adecuado para abordar el problema, puesto que cubre, al menos parcialmente, estas necesidades.

Un flujo de trabajo se define en [60] como: "la automatización de un proceso de negocio, total o parcial, durante la cual documentos, información o tareas son intercambiadas entre los participantes conforme a un conjunto de reglas preestablecidas". En un flujo de trabajo, la información, tareas y documentos pasan de un participante a otro, para que se realicen una serie de acciones de acuerdo con un conjunto de reglas procedimentales; los participantes pueden ser personas o máquinas. Los sistemas que dan soporte a la definición del flujo de trabajo y a su posterior ejecución, se denominan Sistemas de Gestión de Flujos de Trabajo (SGFT).

La comunidad de flujos de trabajo ha detectado hace algún tiempo nuevos retos que hasta el momento no han sido cubiertos adecuadamente por los SGFT [42]. Algunos de ellos están motivados por los avances en la tecnología, tales como la ejecución

distribuida de procesos e interoperabilidad. En cambio, otros provienen de aspectos que han sido sistemáticamente olvidados por los desarrolladores de SGFT, tales como modelado y metamodelado de flujos de trabajo, simulación, análisis de ejecuciones de flujos de trabajo, evolución y reutilización.

Sin embargo, en otras áreas tales como Ingeniería del Software sí se ha abordado problemas de modelado, validación e implementación, como parte del desarrollo de aplicaciones en general. En el área de Extracción de Conocimiento, se ha prestado atención a la búsqueda de información de interés a partir del análisis de datos existentes en las organizaciones. Por lo tanto, a pesar de que el modelado y ejecución de un flujo de trabajo, como disciplina emergente, es relativamente nueva, muchas de las ideas y conceptos que se manejan existen desde hace tiempo y provienen de áreas muy diversas. Además de las anteriores podemos también citar las Bases de Datos, la Ingeniería de Procesos, y otros temas ya más concretos como automatización de trabajos, asignación de rutas a documentos, procesos transaccionales, tratamiento de imágenes, sistemas operativos, computación móvil, etc.

Los SGFT surgieron primero en la industria y más tarde se convirtieron en área de investigación. Esto ha motivado la aparición en el mercado de numerosos productos, catalogados como SGFT, que resuelven problemas concretos pero en los que, en muchas ocasiones, se echa en falta una base científica que permita resolver ciertas limitaciones existentes. Entre ellas podemos citar problemas de escalabilidad, interoperabilidad, robustez, falta de soporte metodológico al modelado y soporte al análisis de registros de ejecución. Todas ellas se detallarán en el siguiente capítulo. Respecto a los productos existentes en el mercado, no todos entran dentro de la misma categoría, ni todos ellos proporcionan las mismas prestaciones a la hora de modelar o ejecutar un flujo de trabajo. Existen diversos tipos de SGFT dependiendo de las características del flujo de trabajo que gestionen. Algunos de estos productos son: MQSeries Workflow de IBM [17], FlowMind [62], Talmia [63], Workflow for ICM [64], Bizflow [1] y ProcessMaker [65].

Finalmente, cabe destacar la organización internacional denominada Workflow Management Coalition (WfMC), cuyo objetivo es aunar esfuerzos para proponer estándares en el campo de los SGFT. Estos estándares proponen un modelo clásico para

la definición de flujos de trabajo, y definen las interfaces de comunicación entre las distintas herramientas que componen el sistema.

2.2 Sistemas de Gestión de Flujos de Trabajo.

Un SGFT es "un sistema que define, crea y gestiona la ejecución de flujos de trabajo mediante el uso de software, siendo capaz de interpretar la definición del proceso, interactuar con los participantes y, siempre que se requiera, invocar el uso de herramientas y aplicaciones" [56].

De acuerdo con la definición anterior, en un SGFT existen dos actividades claramente diferenciadas, aunque con relaciones entre ellas. Por una parte está la definición del flujo de trabajo que implementa al proceso de negocio, en lo que llamaremos modelado del flujo de trabajo, y por otra parte está la animación o ejecución de dicho modelo, también conocido en la literatura con el término inglés *enactment*. La Figura 1 muestra los conceptos básicos y la terminología asociada tanto para la fase de modelado como para la fase de ejecución.

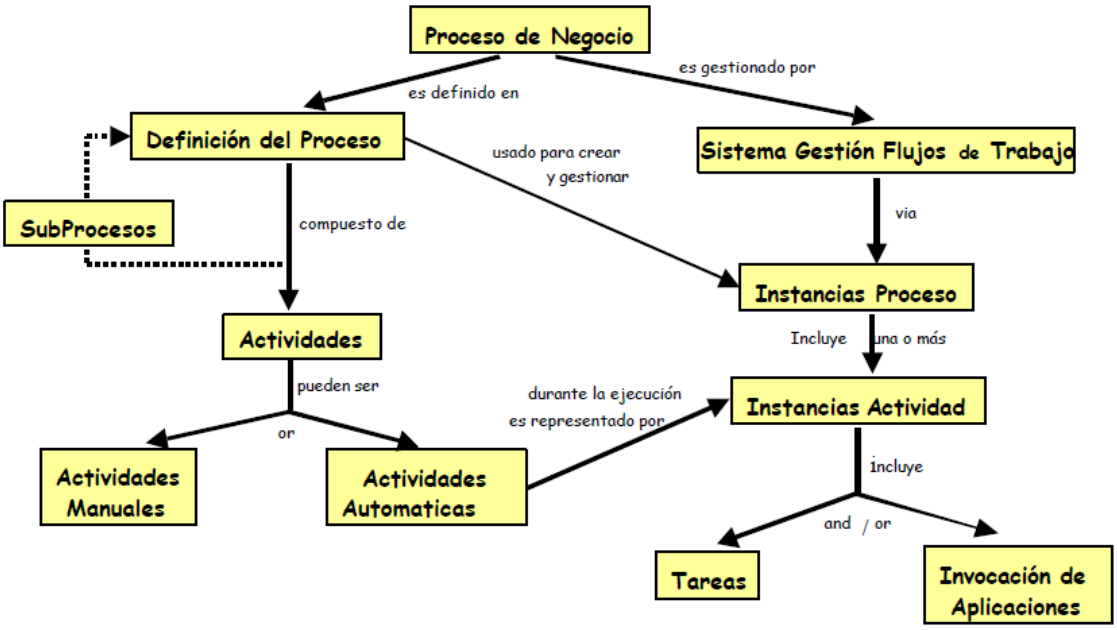


Figura 1. Conceptos básicos y terminología (fuente WfMC)

La especificación del flujo de trabajo consiste en definirlo como un conjunto de actividades relacionadas, más un conjunto de criterios que indican el comienzo y finalización del proceso y de sus componentes, e información adicional sobre cada actividad, tal como participantes, invocación de aplicaciones, datos, etc. Las actividades

describen trozos de trabajo y constituyen pasos o tareas dentro del proceso. Éstas pueden ser manuales o automáticas, en el sentido de que puedan requerir o no recursos humanos para su realización, así como invocar aplicaciones externas que realicen parte o todo el trabajo asignado a dicha actividad.

La ejecución del flujo de trabajo en un SGFT consiste en la creación, manipulación y destrucción de instancias proceso, que representen al flujo de trabajo de acuerdo con la especificación previa. Cada instancia proceso tendrá una identidad visible externamente y un estado interno que representa su progreso hacia la finalización y su estado con respecto a las actividades que lo componen. Cada vez que la ejecución del proceso suponga la invocación de una actividad, según la definición del mismo, se crea una instancia actividad que la representa. Dicha instancia se encarga de ejecutar las acciones asignadas, accediendo a los datos que sean necesarios e invocando la aplicación externa correspondiente, si así lo requiere la actividad.

Los SGFT pueden ser implementados de una gran variedad de formas, como resultado de la utilización de diferentes tecnologías en diferentes entornos, que pueden ir desde un pequeño grupo de trabajo a una gran organización. En cuanto a la tecnología utilizada, son muchas las que pueden confluir en un SGFT. Destacamos gestión de bases de datos, interfaces gráficas de usuario, integración de sistemas (nuevos con los ya existentes), mensajes, gestión de documentos, cliente/servidor y distribución.

2.2.1 Taxonomía

Los elementos a considerar en la especificación de un flujo de trabajo son muchos y de naturaleza muy variada, como ya se ha comentado anteriormente. Por ello, no existe una única clasificación de los SGFT, sino varias dependiendo del criterio a considerar. A continuación se enumeran algunas de las clasificaciones que podemos encontrar en la literatura:

A. Según la complejidad del grafo que representa al flujo de trabajo [14], es decir, de los requisitos de coordinación entre las distintas actividades, tenemos dos tipos de flujos de trabajo:
    - Ligeramente estructurado. En este caso, el grafo que representa la coordinación entre las actividades es prácticamente lineal, y éstas se van ejecutando una a continuación de otra.

- Altamente estructurado. El grafo que representa la coordinación entre las actividades ya no es lineal, sino que representan ejecuciones en paralelo de actividades, sincronización de actividades, etc.

B. Según el grado de participación humana en el flujo de trabajo [14], éstos se pueden clasificar en:
    - Orientados a personas. Cuando la participación humana en la ejecución del SGFT es importante. Estas personas serán agentes que inician o realizan las actividades. Por su naturaleza, la mayor parte de las actividades que conforman el flujo de trabajo son manuales.
    - Orientados a sistemas. La participación humana suele ser menor y el proceso está altamente automatizado, por lo que la mayor parte de las actividades del flujo de trabajo son actividades automáticas.
C. De acuerdo con la tecnología en la que se basa el SGFT, éstos se pueden clasificar en:
    - Centrados en el correo electrónico. En estos SGFT se utiliza el e-mail como medio de comunicación.
    - Centrados en la documentación. La principal característica es que los documentos circulan e interaccionan con aplicaciones externas. Predominan los aspectos de gestión de documentos.
    - Centrados en el proceso. En este caso, lo importante es el propio proceso. La comunicación entre las actividades y los datos se implementan guardando la información en la base de datos, proporcionándose interfaces de interacción.
D. Una clasificación de SGFT bastante aceptada hoy en día, es aquella que distingue entre un SGFT de producción, administrativos, ad-hoc y de colaboración. Esta clasificación se establece en base a dos criterios: la complejidad de las actividades involucradas y la estructura de las mismas, por una parte, y por otra, en base a las similitudes de los procesos de negocio involucrados y su valor o importancia en la propia organización. La Figura 2 y la Figura 3 muestran cómo se sitúa cada tipo de SGFT de acuerdo a los criterios establecidos.
    - Administrativos. Este tipo de SGFT es el que modela los procesos burocráticos de una organización. La función básica asociada al mismo

es el procesamiento de formularios. El SGFT es encarga de activar la ejecución de las actividades (la mayor parte de las cuales son manuales e interviene un agente en ellas), recoger las respuestas (datos) y obtener el formulario (que normalmente es una o varias actividades automáticas, que recopilan y procesan la información obtenida). Por ejemplo, la matrícula universitaria o la obtención de certificados.

La principal característica de las actividades involucradas en el proceso es que se trata de actividades repetitivas y de baja complejidad. La estructura del grafo que representa el proceso puede variar de baja a alta, aunque en muchas ocasiones el grafo de tareas es lineal.

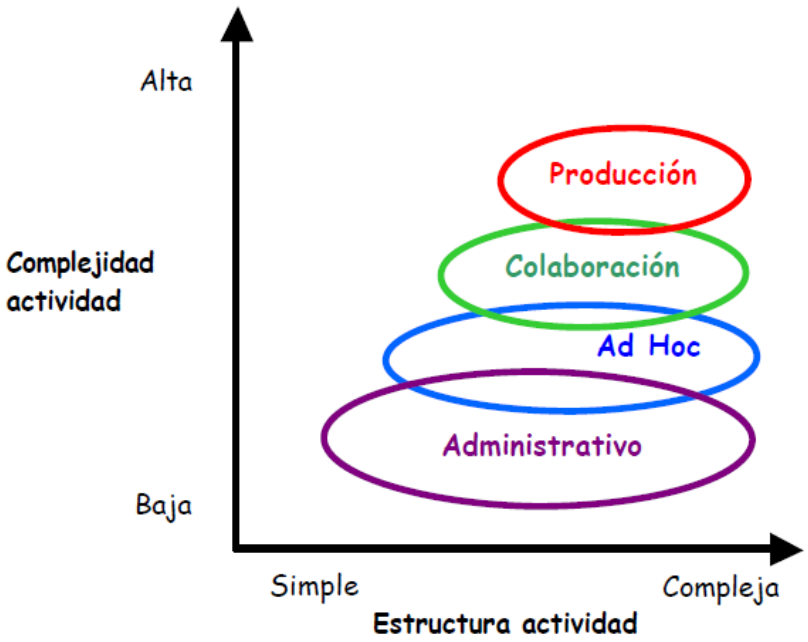


Figura 2. Clasificación de los SGFT según complejidad y estructura de las actividades involucradas

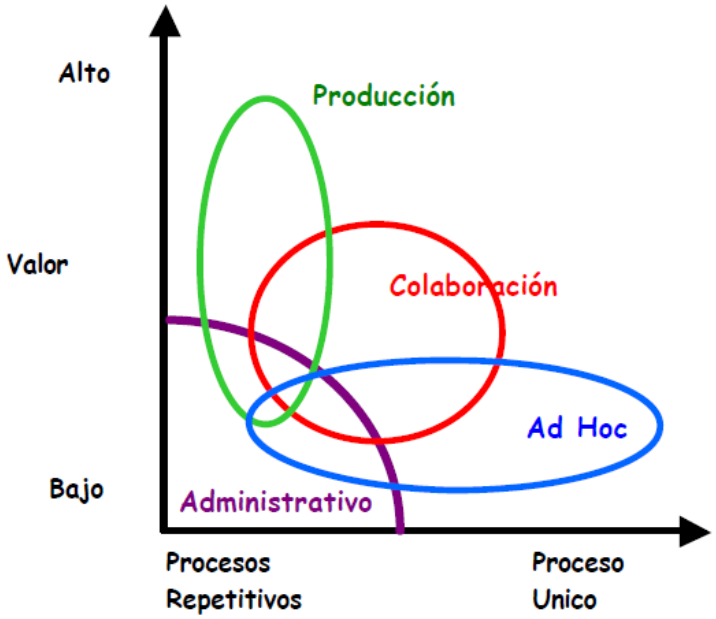


Figura 3. Clasificación de los SGFT según la similitud de los procesos de negocio y su valor en la organización

- Ad-hoc. Estos SGFT son muy similares a los administrativos, pero con la característica de ser utilizados para tratar situaciones únicas o excepcionales, en lugar de procesos burocráticos perfectamente establecidos. El SGFT informa sobre el estado de ejecución de cada actividad y la participación humana es esencial.

  Las actividades involucradas en el proceso suelen ser únicas, en el sentido de que muchas de ellas sólo se realizan una vez y su complejidad oscila de baja a media. La principal dificultad en este tipo de SGFT estriba en construir el grafo de coordinación y cooperación entre actividades.

  Un ejemplo de este tipo es el proceso para la aceptación de artículos en una revista desde el punto de vista del autor. Cada revista tiene su propio proceso a seguir, y en este caso, seguir la pista por la parte del autor de cada proceso de aceptación por separado no es complicado, pero sí en caso de estar todos juntos y ejecutándose simultáneamente (envío simultáneo a varias personas).

- Producción. Modelan e implementan los procesos de negocio críticos de la organización, es decir, los procesos directamente relacionados con la función principal de la misma.

  Los SGFT de este tipo son sistemas complejos que se ejecutan sobre entornos heterogéneos, y en los que suelen participar como agentes gran variedad de personas y organizaciones. Normalmente se requiere la ejecución de transacciones para el acceso a la información, que puede encontrarse en diferentes sistemas. Finalmente, cabe destacar que la monitorización del estado en el que se encuentra cada actividad es muy importante en este tipo de sistemas, así como el posterior análisis estadístico de toda esta información, que va a permitir, entre otras cosas, mejorar el proceso.

  Las actividades involucradas en el proceso son repetitivas (cientos o miles de instancias en ejecución), pero de alta complejidad. La estructura del grafo que representa el proceso suele ser más bien compleja.

  Como ejemplos podemos citar la integración de software preexistente (legacy applications), los préstamos bancarios, préstamos y devoluciones en general, seguros, etc.

- Colaboración. Este tipo de SGFT se caracteriza principalmente por la participación de distintas personas y las interacciones que tienen lugar entre ellas, de tal forma que la mayor parte de la coordinación en realidad la realiza el hombre. Otra característica importante es la existencia de ciertas actividades sobre las que se pueden realizar varias iteraciones hasta que se alcance un cierto consenso por parte de todos los participantes; una vez alcanzado, la ejecución continúa siempre hacia delante. Son sistemas muy dinámicos que en muchas ocasiones se definen conforme se avanza en el proceso.

  Las características enunciadas anteriormente hacen que no exista un consenso total sobre si realmente se trata o no de SGFT, especialmente cuando las características anteriores se llevan al límite, ya que la mayor parte de la coordinación la realiza el hombre y el sistema se limita a

proporcionar un buen interfaz para las interacciones (normalmente vía correo electrónico).

2.2.2 Limitaciones

Las principales limitaciones de los SGFT se pueden resumir en las siguientes:

- Los SGFT existentes no son totalmente compatibles, haciéndose prácticamente imposible unir diferentes sistemas que proporcionan diferentes funcionalidades en uno único. Las incompatibilidades existentes no sólo tienen que ver con la sintaxis o la plataforma en la que se ejecutan, sino que muchas veces dichas incompatibilidades se refieren al propio modelo de ejecución del flujo de trabajo. Esta situación provoca que los diferentes productos existentes se consideren como "islas" de la automatización de procesos.
- En muchas ocasiones, estos productos fueron diseñados para pequeños grupos de usuarios. Cuando estos productos se han querido utilizar a gran escala, han aparecido numerosas restricciones derivadas del propio diseño del producto, tales como pobre soporte a la comunicación, una única base de datos donde se almacena toda la información, etc. Estas restricciones obliga en numerosas ocasiones a rediseñar por completo el SGFT.
- La falta de robustez de muchos productos también es una limitación importante. Ante un fallo, no siempre se aseguran una recuperación correcta del sistema, sino que muchas veces queda en manos de los propios mecanismos de recuperación de la base de datos. En numerosas ocasiones, el problema se debe al igual que antes, al hecho de que se rediseñaron para utilizarse a más pequeña escala, sin tenerse en cuenta qué ocurriría cuando hubieran numerosos componentes (usuarios, aplicaciones y datos) involucrados en la ejecución de un proceso y estos componentes estuvieran distribuidos por la red.
- No existe hasta el momento una formalización de la noción de flujo de trabajo, sino que en la mayoría de las ocasiones simplemente se proporcionan descripciones informales de cuáles son sus componentes.

- No existe soporte metodológico para los procesos de modelado de flujos de trabajo, ni tampoco para el análisis de los mismos [42].

La mayor parte de las limitaciones mencionadas se deben a la falta de un Modelo de Referencia común para todos los SGFT, que sea aceptado como un estándar por parte de todos los desarrolladores, e incorporado en dichos sistemas. Con el objetivo de subsanar esta deficiencia, nace la Workflow Management Coalition (WfMC), cuyos principales resultados se comentan a continuación.

2.2.3 Estandarización

Los principales esfuerzos en el desarrollo y promoción de estándares en el campo de los SGFT han sido realizados por la WfMC. Es una organización internacional, sin ánimo de lucro, que se establece en agosto de 1993 y reúne a un grupo de vendedores, usuarios, analistas y grupos de investigación. Su principal objetivo es promover el uso de SGFT mediante el establecimiento de estándares que faciliten la creación, desarrollo y análisis de estos sistemas.

La WfMC ha definido una serie de estándares (ver Figura 4). De entre todos ellos, destacamos el Modelo de Referencia [16] que describe la arquitectura básica de un SGFT y que junto con el de Terminología y Glosario [55], constituyen el material básico. En base a ellos se han definido otras especificaciones con una finalidad mucho más concreta, como son: interoperabilidad entre SGFT [56], modelo para la definición de procesos [52], invocación a través de API (Aplication Programming Interface) [53] y análisis de datos [54]. En las siguientes secciones se comentan más extensamente el modelo de referencia propuesto y se darán las principales características del resto.

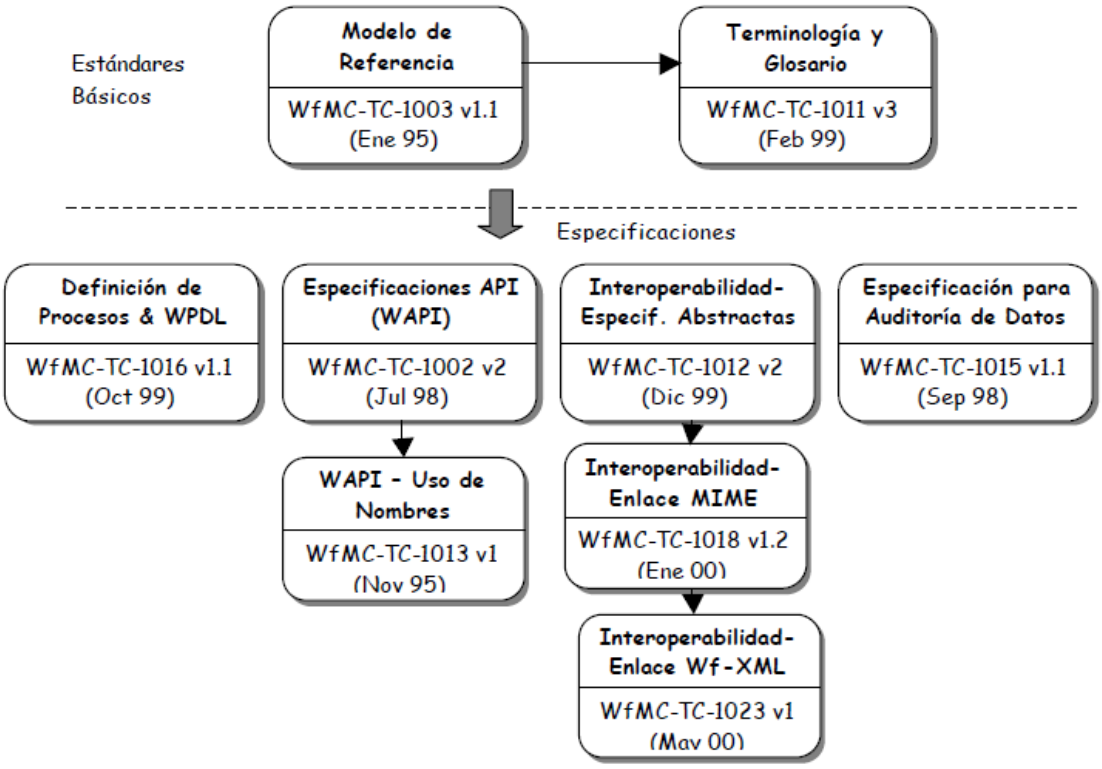


Figura 4. Estándares de la WfMC

Posteriormente, Object Management Group (OMG) también une sus esfuerzos con la WfMC en la propuesta de estándares. OMG es una organización internacional sin ánimo de lucro fundada en 1989, cuyo principal objetivo se centra en promover la teoría y práctica de la tecnología orientada a objetos en el desarrollo de software. Entre sus principales propuestas está la arquitectura OMA (Object Management Architecture), que establece el marco conceptual en el que se basan todas las especificaciones, y CORBA (Common Object Request Broker Architerture) como respuesta a las necesidades de interoperabilidad entre el gran número de productos software y hardware existentes hoy en día, permitiéndose la comunicación entre ellos. En 1997 se inicia un acercamiento hacia los SGFT, publicando en [29] una propuesta para incluir funcionalidad de ejecución de los SGFT en entornos desarrollados siguiendo una arquitectura OMA, lo cual mejoraría las prestaciones de dichos entornos.

Este acercamiento de OMG conlleva la publicación de una primera especificación, basada en los estándares de la WfMC, que establece requisitos de interoperabilidad entre distintos SGFT en un entorno CORBA [30]. A partir de ese momento, se inicia un proceso de revisión y mejora de esta especificación, siendo [31] la última propuesta.

2.2.3.1 Modelo de Referencia de la WfMC

El modelo de Referencia propuesto por la WfMC intenta reunir las características comunes de cualquier producto para la gestión de flujos de trabajo, de manera que sea posible la interoperabilidad entre ellos, a través de estándares comunes para cada una de las funciones que se puedan realizar. En primer lugar, se han identificado las distintas áreas funcionales y a continuación se han desarrollado especificaciones para la implementación de las mismas, asegurándose la interoperabilidad entre ellos, a través de estándares comunes para cada una de las funciones que se puedan realizar. En primer lugar, se han identificado las distintas áreas funcionales y a continuación se han desarrollado especificaciones para la implementación de las mismas, asegurándose la interoperabilidad entre distintos SGFT y su integración con otras aplicaciones informáticas.

Este modelo ha sido desarrollado a partir de la arquitectura de una aplicación de sistema de flujo de trabajo genérica, identificando sus componentes y las diferentes interfaces que permiten comunicación a diferentes niveles (Figura 5)

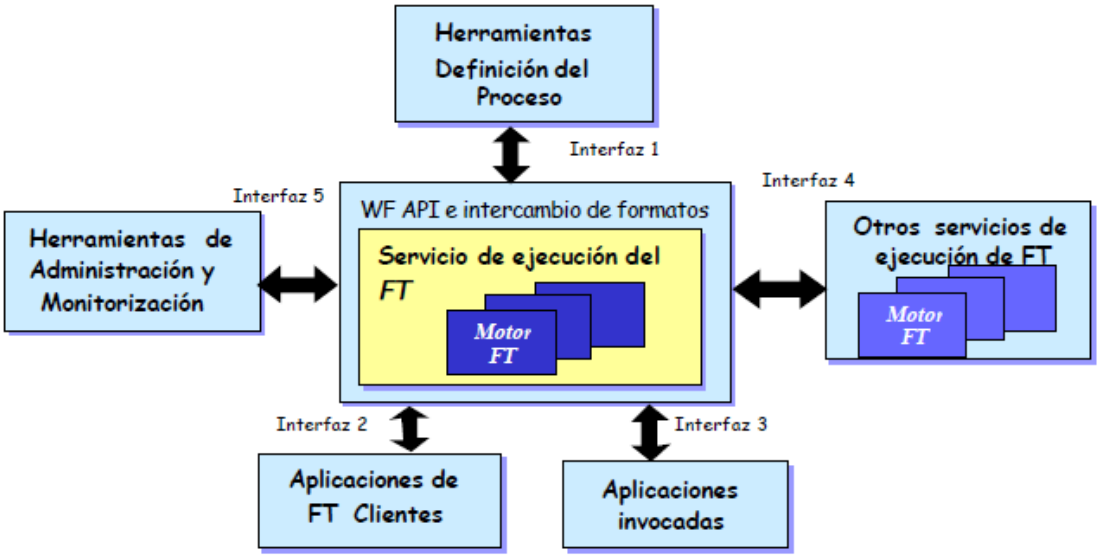


Figura 5 El modelo de referencia de un SGFT

El componente central es el servicio de ejecución del flujo de trabajo que se encarga de crear, gestionar y ejecutar cada una de las instancias del modelo de flujo de trabajo. En este componente es donde se encuentra el motor del SGFT, que proporciona la ejecución, propiamente dicha, de cada instancia. En caso de estar en un entorno de ejecución de flujo de trabajo distribuido, pueden existir diferentes motores de flujo de

trabajo que controlen distintas partes de la ejecución del proceso. La comunicación de este componente con el resto se realiza a través de lo que la WfMC denomina WAPI (Workflow APIs), es decir, la interfaz para la programación de aplicaciones de flujo de trabajo [53].

Otro de los componentes son las herramientas de definición del proceso o flujo de trabajo, las cuales permiten modelar, describir y documentar un determinado flujo de trabajo, proceso de negocio o proceso en general. Estas herramientas pueden ser totalmente informales (lenguaje natural, lápiz y papel) o mucho más sofisticadas y formalizadas (interfaces gráficas con un modelo subyacente bien definido). Se debe especificar la lógica del proceso, las actividades que lo componen, los participantes humanos, aplicaciones invocadas, datos utilizados, etc. La interfaz 1 permite la comunicación entre este componente y el servicio de ejecución del flujo de trabajo.

La distinción de ambo componentes aporta claros beneficios, entre los que destaca la separación entre el entorno de ejecución y el de definición. De este modo, es posible almacenar en un repositorio la información sobre la definición del proceso y ésta ser accedida por distintos entornos de ejecución para ejecutarlo completamente o de forma distribuida. La interfaz 1 se encarga por tanto del intercambio de información entre el componente que permite la definición del proceso y el propio servicio de ejecución del flujo de trabajo. Esta interfaz hace necesaria la definición de un metamodelo básico, en el que se identifica el conjunto mínimo de entidades para la definición de un proceso, permitiendo el intercambio de información entre ambos componentes.

Además del propio servicio de ejecución del flujo de trabajo y las herramientas para la definición del proceso, tenemos otro componente denominado aplicaciones clientes del flujo de trabajo, que representa las entidades software utilizadas por el usuario final en aquellas actividades que requieren participación humana para su realización. Si este componente se separa de lo que es el propio componente de ejecución, es necesaria una interfaz (interfaz 2) que defina y maneje claramente el concepto de lista de trabajos o worklist, como una cola de trabajo asignado a un usuario o a un grupo de usuarios por el propio motor de ejecución del flujo de trabajo.

El componente aplicaciones invocadas representa software o aplicaciones ya existentes que un SGFT puede utilizar para la realización de ciertas actividades, teniendo en cuenta que, en principio, dichas aplicaciones se pueden encontrar en cualquier

plataforma o lugar de la red. La interfaz 3 permite la comunicación entre este componente y el servicio de ejecución del flujo de trabajo, no sólo a nivel de invocación del mismo, sino de transformación de datos en formatos entendibles por ambos componentes. Una posible solución se obtiene a través de lo que se denomina agente aplicación, de modo que el servicio de ejecución del flujo de trabajo se comunica con las funciones estándar de dicho agente aplicación y éste define interfaces específicas para cada tipo de aplicación invocada.

La interoperabilidad entre SGFT está representada por el componente denominado otros servicios de ejecución de flujo de trabajo, siendo la interfaz 4 la que permite dicha comunicación. En este caso, la WfMC ha desarrollado un conjunto de escenarios de interoperabilidad que van desde la conexión a nivel de actividades simples hasta todo un completo intercambio de definición de procesos y datos [61].

Finalmente, el componente herramientas de administración y monitorización permite que distintos servicios de ejecución de flujo de trabajo compartan las mismas funciones de administración y monitorización del sistema, como pueden ser, por ejemplo, la gestión de usuarios, el control de los recursos y la supervisión del estado de todo el proceso.

2.3 Conclusiones

En este capítulo se han presentado las principales características de los SGFT, desde el punto de vista de su funcionalidad, detectándose que existen dos actividades básicas: definición y ejecución. La taxonomía presentada demuestra que existe gran variedad dentro de los SGFT, y que a pesar de que todos ellos permiten la automatización de procesos, siguen existiendo limitaciones importantes, relacionadas principalmente con la falta de un soporte metodológico y la falta de estándares que sean aceptados e implementados por los sistemas comerciales.

Precisamente, con éste último objetivo nace la WfMC, una organización internacional que está trabajando junto con OMG para la definición de estándares y su aceptación. De entre todos ellos, destaca el Modelo de Referencia, que establece un marco común a partir del cual distintos SGFT pueden interactuar. Esto determina la aparición de toda una familia de estándares para aspectos mucho más concretos y relacionados principalmente con temas de interoperabilidad. En los últimos años han ido apareciendo también propuestas relativas a la definición de flujos de trabajo, pero principalmente

relacionadas con el establecimiento de formatos de intercambio de modelos de flujo de trabajo, para su exportación a otros SGFT, y no a nivel metodológico para la obtención y validación de dicho modelo.

Para terminar el capítulo y según la taxonomía vista anteriormente, podemos decir que el motor de workflows que se ha definido mediante esta Tesis, tiene las siguientes características:

- Altamente estructurado
- Orientado a las personas
- Centrado en el proceso
- Colaborativo

Particularmente, esta última característica es la que ha llevado a la realización de este trabajo, es decir, el desarrollo de un motor de workflows flexibles en el que las personas son los protagonistas.

Capítulo 3. Modelos de Flujos de Trabajo

El contenido de este apartado ha sido extraído de la tesis doctoral de J. Fernández [59].

3.1 Representación de Flujos de Trabajo

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

Existen innumerables modelos y lenguajes de representación de WF que pueden ser utilizados como herramientas para diseñar procesos de negocio como BPMN [28] o XPDL [52] que serán analizados más adelante. Los sistemas de gestión de WF comerciales suelen venir acompañados de completas utilidades gráficas para ayudar al máximo al experto a diseñar sus procesos. Estas herramientas gráficas suelen potenciar la facilidad de la descripción de los WF ya que facilitan la legibilidad y hacen más entendible el sistema. El mayor problema de las herramientas comerciales es que suelen pecar de estar demasiado pensadas para solucionar los problemas de ejecución de WF más que los problemas de representación por lo que pecan de falta de expresividad. Por otro lado, aunque también hay modelos más orientados a resolver el problema de la representación, que pueden venir en forma de estándares o iniciativas individuales que intentan aportar solución a problemas generales o específicos de distintos entornos, estos suelen tener problemas a la hora de la ejecución, ya que estos lenguajes suelen ser más difíciles de interpretar debido a su complejidad.

3.1.1 Características de los lenguajes de representación de Flujos de Trabajo

Para representar un WF, hay que definir un lenguaje capaz de expresar todas las situaciones que requiera el problema a resolver, de la forma más entendible posible. El amplio ámbito de aplicación de los lenguajes de representación de WF hace que unos modelos puedan ser válidos en unos entornos e inválidos en otros. Por ejemplo,

entornos que necesiten una gran legibilidad, pueden restringir la expresividad para conseguir un lenguaje válido.

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

3.1.1.1 Expresividad

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

Existe una iniciativa [48] para la creación y el mantenimiento de distintos patrones de comportamiento que podrían ser incorporados en la definición de WF. Esta iniciativa, parte de la base publicada por Will van der Aalst en [50] donde se publicaron los 20 patrones básicos que todo Sistema de Gestión de Workflows debería optar a cumplir. Estos patrones se han convertido en una medida de la expresividad de los modelos de representación de WF, de modo que cuantos más patrones sea capaz de expresar un lenguaje, más expresivo será. Esta iniciativa, no solo crea, mantiene y revisa estos patrones, sino que actúa de observatorio sobre las herramientas de gestión de WF activas en el mercado, realizando una evaluación continua de estas.

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

- **Patrones de Control de Flujo.** Estos patrones se corresponden con una revisión de los patrones de control de flujo presentados por Will van der Aalst en [25]. Estos patrones definen situaciones de gestión del flujo de procesos de negocio,

tales como sincronizaciones o separación paralela de procesos. En la Figura 6 se puede ver un ejemplo de WF en el que se pueden identificar varios patrones de control de flujo, entre ellos un patrón de separación paralela, donde los procesos 1 y 2 se ejecutan paralelamente después de ejecutarse el proceso 0; y, análogamente, un patrón de sincronización paralela donde después de ejecutar los procesos 3 y 4 se ejecuta el proceso 5.

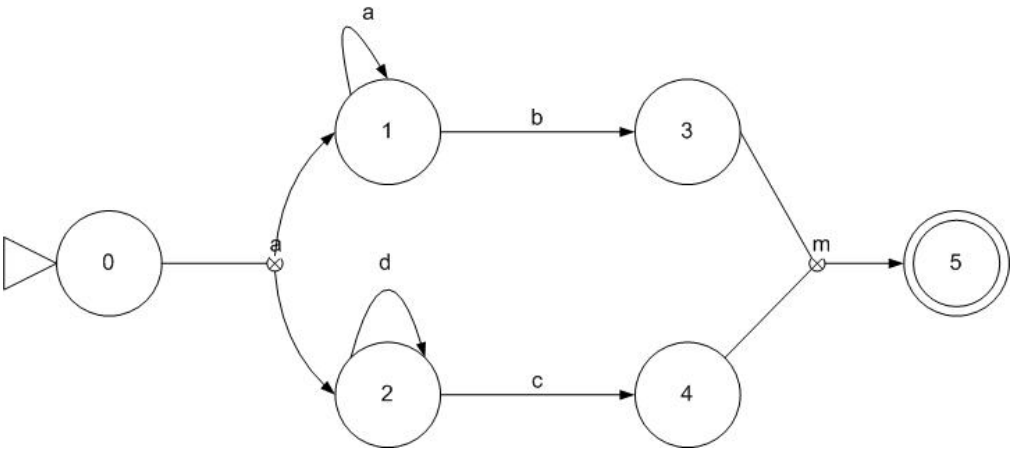


Figura 6. Ejemplo de Workflow con patrones de control de flujo

- **Patrones de Datos.** Presentados en [23], estos patrones ofrecen una serie de conceptos que pueden aplicarse a la utilización de datos en los sistemas de WF. Estos patrones no solo definen la forma en la que los datos pueden emplearse dentro de los WF y los datos que los sistemas de gestión de WF puede usar, sino que también caracterizan la interacción de los elementos de datos con otros WF o sistemas externos. En la Figura 7 se presenta un ejemplo de patrón de WF de datos, en él se presenta una rutina basada en datos, por la cual, la transición al siguiente estado depende de operaciones sobre los datos; ya estén estos localizados en el repositorio de datos del motor de WF, en el de la instancia de WF, o incluso a la salida del proceso ejecutado. En la Figura, se puede observar como el paso de la acción A a las acciones B, C y D, está supeditado a variables dependientes de la instancia en ejecución (N) o incluso a variables globales a todas las ejecuciones de WF como M.

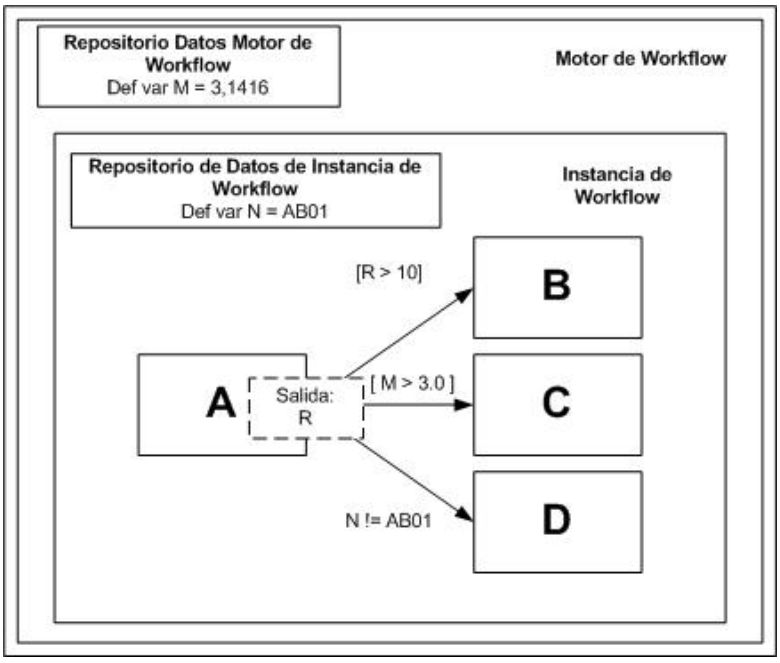


Figura 7. Ejemplo de Workflow con patrones de datos

- **Patrones de Recursos.** Estos patrones [24] están centrados en el modelado de recursos y su interacción con sistemas de gestión de procesos de negocio. Estos recursos pueden ser humanos, o no humanos, como equipamientos, salas, etc. En el ejemplo del proceso de ventas, cuando se producía un pedido, desde el departamento de ventas se pedía un informe de morosidad. En este caso, el responsable del departamento de contabilidad podría ser tratado desde el WF como un recurso compartido, que requeriría un control que gestione el acceso a este. Este tipo de control de los recursos, son los estudiados por el conjunto de los patrones de recursos.
- **Patrones de Manejo de Excepciones.** Estos patrones definen soluciones al manejo de las excepciones ocurridas durante el la ejecución de WF. Estos patrones fueron presentados en [26]. En el ejemplo del proceso de ventas si el formulario de petición del informe de morosidad está incompleto o es erróneo, se producirá una excepción en el proceso. El protocolo de acciones a realizar en caso de que esto se produzca, es el ámbito de aplicación los patrones de manejo de excepciones.

Los Patrones de Control de Flujo, son los patrones más usados para medir la expresividad de los modelos, esto es debido, a que son los que definen la coordinación

de las acciones, mientras que los demás describen situaciones de bajo nivel más dependientes de la implementación de los procesos que del flujo en sí.

En este trabajo únicamente al flujo que las acciones siguen. No se van a abordar por tanto ni el ámbito de las variables de paso, ni el control de los recursos, ni el manejo de las excepciones que no están explícitamente representadas en el proceso en sí. Por ello, en este trabajo se van a utilizar solo los Patrones de Control de Flujo para medir la expresividad. En el Anexo A se describen con más detalle los Patrones de Control de Flujo que serán utilizados para la evaluación de los modelos.

3.1.1.2 Legibilidad

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

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

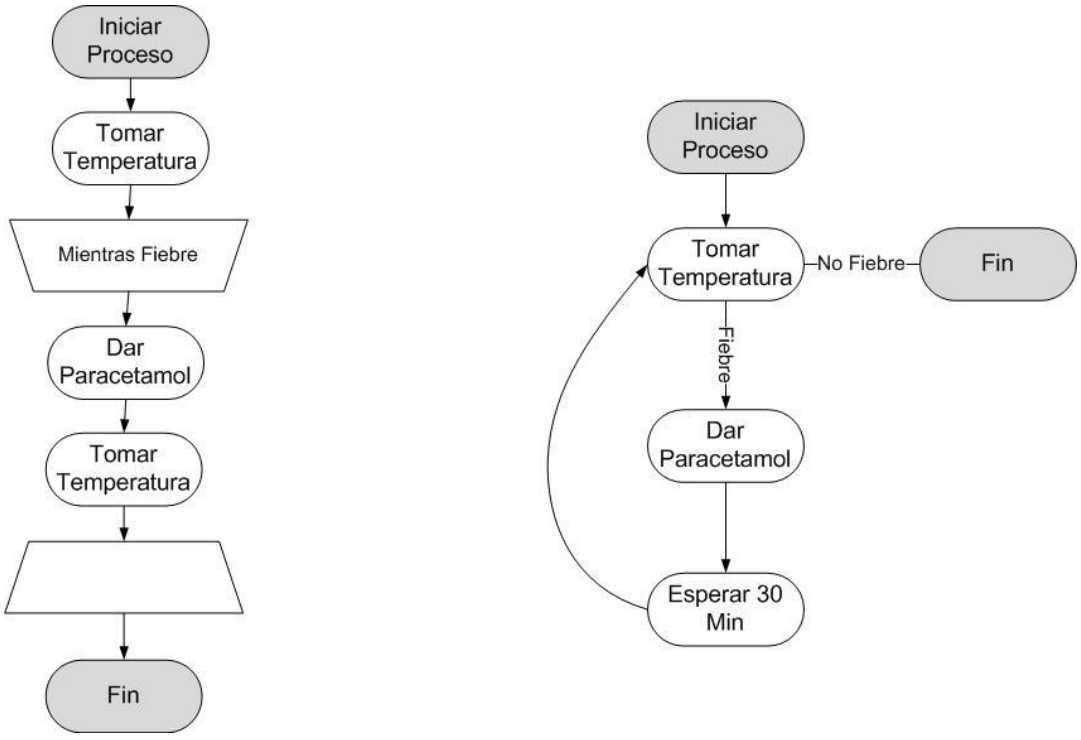


Figura 8. Representación de un Workflow con estructuras de programación y con Autómata Finito

La legibilidad de un WF es un valor muy subjetivo. Modelos que suelen ser legibles para programadores informáticos no suelen serlo para expertos externos. Por ejemplo, en la Figura 8 se muestra un proceso diseñado de dos formas diferentes. La primera se realiza utilizando estructuras de flujo complejo como la estructura de programación mientras, por otro lado la segunda, solo utiliza estados y flechas. Entre estos dos WF un programador informático consideraría más legible la primera, mientras que un médico puede sentirse más cómodo con modelos más parecidos al de la segunda.

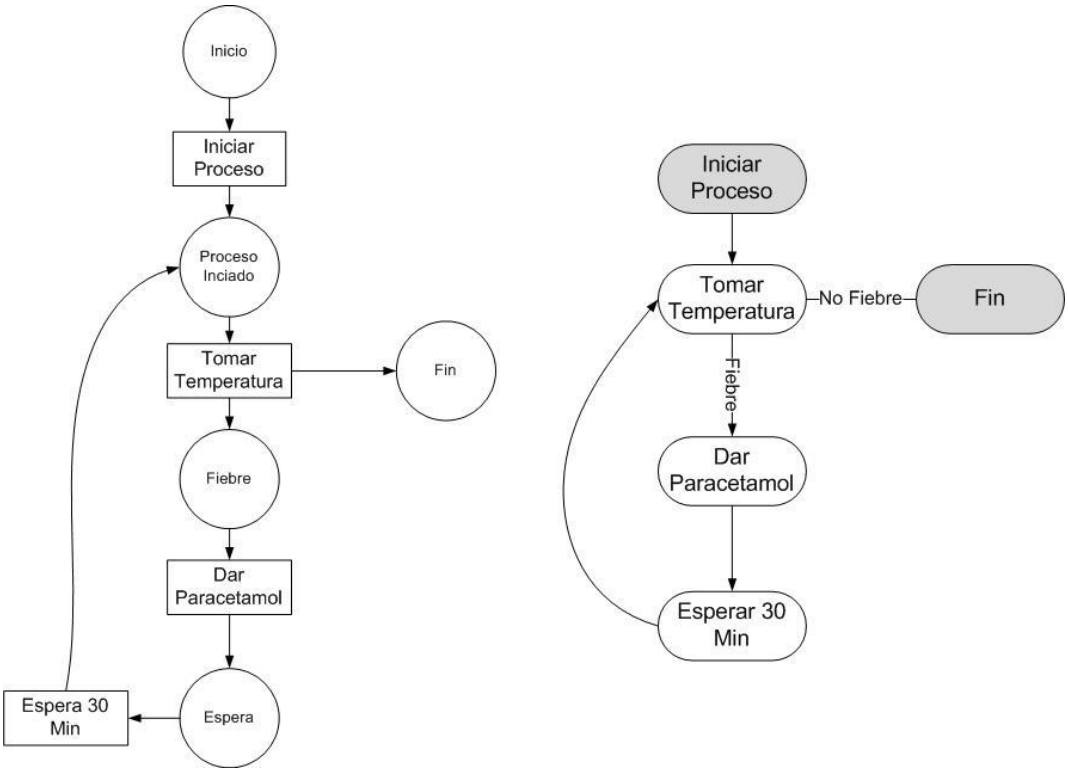


Figura 9. Representación de un Workflow con Redes de Petri y con Autómata Finito

Dejando a un lado estas diferencias subjetivas, otro factor que afecta sobre la legibilidad es la talla del modelo. Si un modelo utiliza más estructuras que otro para representar el mismo proceso, esto hace que su legibilidad se vea reducida. Por ejemplo, en la Figura 9 se ve el ejemplo anterior representado con una Red de Petri. La Red de Petri (que no se va a explicar en este documento) es mucho más expresiva que los autómatas finitos, sin embargo, su legibilidad es menor porque requiere más estructuras para definir los procesos.

3.2 Interpretación de Flujos de Trabajo

Una vez definido un lenguaje de Representación de WF adecuado el siguiente paso es su interpretación. Interpretar un WF es ejecutar las acciones del WF según las reglas

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

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

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

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

3.2.1 Representación e Interpretación

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

Usualmente los modelos de interpretación de WF se presentan definiendo una arquitectura para diseño y ejecución de procesos separado en capas. En la Figura 9 se presenta un esquema representativo del funcionamiento general de un sistema de Interpretación de WF. Cada uno de los elementos de la arquitectura se presenta a continuación:

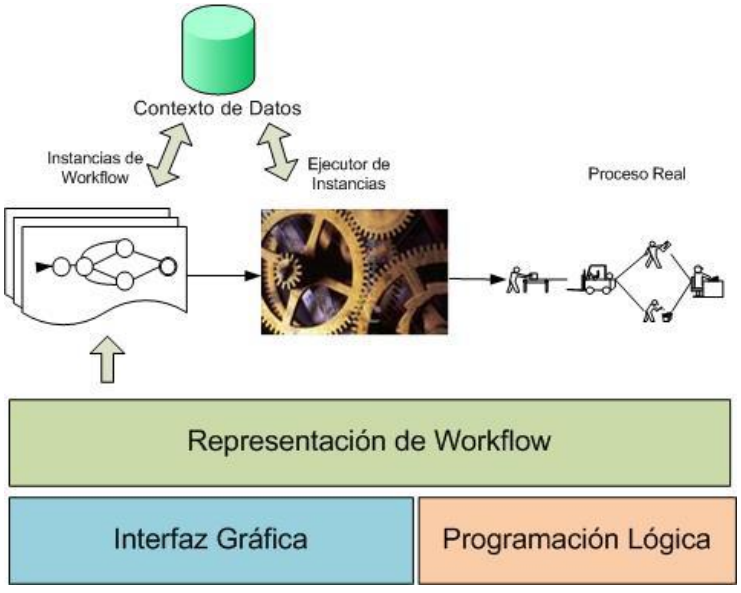


Figura 9. Arquitectura general de interpretación de Workflows

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

familiarizados en el conocimiento de la programación el diseño procesos basados en estos sistemas.

- **Capa de Programación Lógica.** Como ya se expuso anteriormente, la representación gráfica de WF tiene la ventaja que permite una definición de WF sencilla al alcance de cualquier experto en procesos de negocio que no tenga nociones de programación. Sin embargo, estas definiciones, distan mucho de las necesarias para la creación de sistemas formales para la especificación de WF. La capa de Programación Lógica está pensada para tratar con este problema. En esta capa se definen parámetros de configuración y las primitivas que serán usadas por la capa de interfaz gráfica para definir los WF.
- **Capa de Representación de WF.** Esta capa se encarga de juntar la información gráfica de WF, junto con la información parametrizada de este, para especificar formalmente WF que puedan ser ejecutados. Los Diseñadores de procesos dibujan los WF utilizando las librerías gráficas existentes en la capa de interfaz gráfica, que pueden estar asociadas a librerías de ejecución de acciones situadas en la capa de Programación lógica. Por ejemplo, una de las acciones de la capa de interfaz gráfica podría ser enviar un SMS, la capa de programación lógica contendría las librerías necesarias para enviar el mensaje. Por otro lado, la capa de interfaz gráfica cuenta con iconos gráficos que representan el envío de mensajes SMS. Estos iconos son incorporados a la descripción del WF diseñando el proceso de una forma visual.
- **Ejecutor de Instancias de WF.** Esta capa es el núcleo de ejecución de los procesos. Cada una de los WF especificados en la capa de representación son traducidos a un lenguaje sencillo preparado para ser interpretado que y posteriormente ejecutados por el ejecutor de instancias tras ser instanciados. Este elemento es el que finalmente coordina la ejecución ordenada de las acciones según está marcada por las reglas definidas en el WF.
- **Capa de Contexto de Datos.** Esta capa almacena los datos que las instancias de WF generan en tiempo de ejecución. Esta capa es útil para permitir la compartición de datos entre instancias así como la gestión de las variables de entrada y salida que requieran los WF.

Este modelo general de representación e interpretación de WF puede sufrir cambios debido al campo de aplicación de los WF. Por ejemplo, en entornos donde el diseño de

WF no va a ser realizado por expertos externos sino que es realizado por profesionales informáticos, la capa de interfaz gráfica puede ser eliminada. En este caso los programadores definen los WF utilizando directamente lenguajes formales orientados a la interpretación de procesos.

3.2.2 Problemas de la interpretación de Flujos de Trabajo

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

3.2.2.1 Diseño expresivo, sencillo y legible

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

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

3.2.2.2 Comunicación con procesos externos

Muchas de las actividades a realizar dentro de un WF pueden necesitar acceso a procesos externos nativos o distribuidos por la red para poder llevar a cabo sus

funciones. Acceso a bases de datos, comunicación con aplicaciones nativas o acceso a Web Services son algunos de los ejemplos de lo que un sistema de gestión de WF puede necesitar para ejecutar las actividades que tenga programadas. Este problema suele ser solucionado por los sistemas software permitiendo la incorporación de código nativo en la especificación del WF. No obstante, dentro de este marco, se han desarrollado arquitecturas software orientadas a la solución de este tipo de problemas promoviendo la utilización de herramientas software de comunicación que permiten la intercomunicación de aplicaciones heterogéneas, este tipo de utilidades se denominan Motores de Integración o Enterprise Service Buses (ESB) [9]. Los principales motores de interpretación del mercado cuentan con implementaciones de estos buses para facilitar la interconexión de los sistemas legados con las nuevas aplicaciones.

3.2.2.3 Monitorización de instancias de Flujos de Trabajo

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

3.2.2.4 Modificación de Instancias de Flujos de Trabajo

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

Modificar la instancia obliga a cancelar el proceso de automatización y personalizar manualmente el proceso. Esto puede significar diseñar completamente un WF solo para

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

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

3.3 Selección de un Sistema de gestión de Flujos de Trabajo

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

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

3.3.1 TUNE-UP

La metodología que se presenta en este documento no se caracteriza por ser ni muy representativo ni muy interpretativo, como podemos ver en las tablas 4 y 5 donde el número de patrones que acepta no es muy elevado. En cambio posee otras características que la hacen ser una opción a considerar como se explicará en un capítulo posterior. Alguna de esas características es, la flexibilidad, ya que es posible cambiar el cambio del Flujo de Trabajo de una instancia en ejecución, además de poder

de poder acceder a cualquier punto del Flujo de Trabajo y ver en cualquier momento el estado actual de la instancia en ejecución.

3.3.2 Modelos Orientados a la Representación

Se han definido modelos específicos para representar WF. Dentro de estos están aquellos que han sido diseñados para ser fácilmente entendibles por expertos y que usualmente utilizan metáforas gráficas para describir la ejecución de los procesos. Estos modelos han sido tradicionalmente pensados para formar parte de las especificaciones de requisitos de los sistemas software [39] ya que ayudan a recoger el conocimiento del usuario de una forma fácilmente entendible y reproducible.

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

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

Código	**Patrón**
WPC-1	Secuencia
WPC-2	Separación Paralela
WPC-3	Sincronización
WPC-4	Elección Exclusiva
WPC-5	Fusión Simple
WPC-6	Multielección
WPC-7	Fusión Sincronizada Estructurada
WPC-8	Multifusión
WPC-9	Discriminador Estructurado
WPC-10	Ciclos Arbitrarios
WPC-11	Terminación Implícita
WPC-12	Múltiples instancias sin sincronización
WPC-13	Múltiples instancias con conocimiento en tiempo de diseño
WPC-14	Múltiples instancias con conocimiento en tiempo de ejecución
WPC-15	Múltiples instancias sin conocimiento previo

WPC-16	Elección Aplazada
WPC-17	Rutina Paralela Entrelazada
WPC-18	Hito
WPC-19	Cancelación de Actividad
WPC-20	Cancelación de Instancia

Tabla 1. Listado de Patrones de control de WF usados para evaluar la expresividad de los modelos

3.3.2.1 BPMN

En cuanto a la expresividad, este lenguaje cumple bastantes de los patrones de WF de flujo de control definidos por Van der Aalst en [50] que pueden verse en la Tabla 1. Sin embargo, tiene problemas para expresar los patrones de discriminación estructurada (WPC-9), rutina paralela entrelazada (WPC-17) y los patrones de hito (WPC-18).

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

Este estándar se explicará en detalle en el siguiente capítulo.

3.3.2.2 XPDL

El lenguaje XPDL [52] (del inglés, XML Process Definition Language) es un estándar de la Workflow Management Coalition. Este modelo fue inicialmente diseñado para intercambiar el diseño de procesos de negocio entre distintas herramientas. En este lenguaje no solo se especifica la coordinación entre procesos, sino que también almacena el tamaño y las coordenadas X e Y de los ítems definidos.

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

El lenguaje XPDL es un lenguaje que no solo provee de un conjunto de metáforas gráficas para el diseño de procesos, sino que también provee de un formato de fichero estándar basado en XML. XPDL está específicamente pensado para almacenar e intercambiar el diagrama de procesos. Sin embargo, XPDL no cuenta con un sistema

estándar de Interpretación de WF. Algunos sistemas de interpretación de WF como Enhydra Shark [13], Bonita [19] o WfMOpen [49] utilizan XPDL como lenguaje de representación de WF. La principal desventaja de XPDL es que no está pensado para ser ejecutado, con lo que es necesario añadir información específica no estándar al lenguaje que hace que los distintos sistemas de interpretación no sean compatibles entre sí.

3.3.3 Modelos orientados a la interpretación de Flujos de Trabajo

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

3.3.3.1 BPEL o WSBPEL

El lenguaje BPEL [57] (del inglés, Business Process Execution Language) es un lenguaje de orquestación de procesos definido por el comité de estandarización OASIS. BPEL nació como combinación de los lenguajes WSFL de IBM y XLANG de Microsoft. BPEL fue un acuerdo entre estas dos empresas para definir un lenguaje común que permitiera realizar conexiones entre las aplicaciones creadas tanto por productos de Microsoft como de IBM.

BPEL es un lenguaje eminentemente comercial serializado en XML cuya misión principal es definir procesos de negocio que interactúen con entidades externas a través de Web Services [57]. Por eso, BPEL es también conocido como WSBPEL. BPEL es un lenguaje pensado para ser ejecutado, por lo que no tiene una notación gráfica para el diseño de procesos. BPEL cuenta con la ventaja de la formalidad en la representación de la orquestación de procesos, ya que es un lenguaje que se ejecuta directamente por lo que no admite ambigüedad.

En cuanto a la capacidad de representación de los Patrones de WF, BPEL representa menos patrones que los modelos orientados a la Representación. Entre las limitaciones de BPEL en cuanto a la expresividad cabe destacar que no puede representar patrones de MultiFusión (WPC-8), es decir, no permite la fusión de ramas heterogéneas, patrones

de Ciclos Arbitrarios (WPC-10), por lo que no permite la definición de ciclos de cualquier tipo y ni la definición de hitos (WPC-18), es decir, que impide representar transiciones dependientes de hitos alcanzados.

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

3.3.3.2 jBPM

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

jBPM tiene implementado un motor de ejecución capaz de ejecutar los WF diseñados tanto por la herramienta gráfica, como por otras herramientas capaces de generar código BPEL. jBPM está orientado a su utilización como orquestador de Web Services, aunque puede trabajar también como orquestador de procesos codificados en Java. Debido a la necesidad de una ejecución eficiente, el equipo de desarrollo de jBPM en su implementación actual, ha optado por la creación de un traductor de WF a código nativo para la ejecución. Esto, que hace que el código sea mucho más eficiente que la interpretación directa de los WF, pero hace que no sea posible la modificación de WF en tiempo dinámico. Usando jBPM, la única solución posible es cancelar el WF actual y ejecutar otro de nuevo.

Este motor de workflows se explicará en detalle en el siguiente capítulo 5.

3.3.3.3 Windows Workflow Foundation

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

de estados y modelos de flujo, no siempre es posible ejecutar todas las combinaciones de patrones posibles.

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

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

Este motor de workflows se explicará en detalle en el siguiente capítulo 5.

Capítulo 4. Business Process Modeling Notation

4.1 ¿Qué es BPMN?

El contenido de este apartado ha sido extraído del artículo de introducción a BPMN por Stephen A. White [44].

La notación Business Process Modeling Notation (BPMN) ha sido desarrollada por el Business Process Management Initiative (BPMI) con el objetivo dar una notación rápidamente comprensible por toda la gente de negocios, desde el analista que hace el borrador inicial de los procesos, pasando por los desarrolladores técnicos responsables de implementar la tecnología que llevarán a cabo dichos procesos, llegando finalmente a la gente de negocio que gestionará y monitorizará esos procesos. Las versiones por las que ha pasado dicha notación son: versión 1.0 salió al público en mayo de 2004, versión 1.1 en febrero de 2008, versión 1.2 en enero de 2009 y versión 2.0 beta en septiembre de 2009.

Además, BPMN está apoyado en un modelo interno que genera el ejecutable BPEL4WS. Así, BPMN crea un puente estandarizado para el hueco entre el diseño de los procesos de negocio y la implementación de procesos.

BPMN define un Business Process Diagram (BPD), que se basa en una técnica de grafos de flujo para crear modelos gráficos de operaciones de procesos de negocio. Un modelo de procesos de negocio, es una red de objetos gráficos, que son actividades (trabajo) y controles de flujo que definen su orden de rendimiento.

4.2 Fundamentos de BPMN

Un BPD está formado por un conjunto de elementos gráficos. Estos elementos habilitan el fácil desarrollo de diagramas simples que serán familiares para la mayoría de analistas de negocio (diagrama de flujo). Uno de los objetivos del desarrollo de BPMN es crear un mecanismo simple para crear modelos de procesos de negocio, y al mismo tiempo que sea posible gestionar la complejidad inherente en dichos procesos. El método elegido para manejar estos dos conflictivos requisitos fue organizar los aspectos gráficos de la notación en categorías específicas. Esto da un pequeño grupo categorías que alguien que lea un BPD pueda reconocer fácilmente los tipos básicos de elementos y pueda entender el diagrama. Dentro de las categorías básicas de elementos, se puede añadir información y variaciones adicionales para dar soporte a los requerimientos

complejos sin cambiar dramáticamente el "aspecto" básico del diagrama. Las cuatro categorías básicas de elementos son:

- Objetos de flujo
- Objetos conectores
- Artefactos
- Swimlanes

4.3 Objetos de flujo

Un BPD es un pequeño conjunto (tres) de elementos básicos, que son los *Objetos de Flujo*. Los tres objetos de flujo son:

- **Evento:** un evento se representa con un círculo. Es algo que "pasa" durante el curso del proceso de negocio. Estos eventos afectan al flujo del proceso y suelen tener una causa (trigger) o un impacto (resultado). Los eventos representados con un círculo con centro abierto permiten a los marcadores internos diferenciar diferentes triggers y resultados. Hay tres tipos de eventos, como se muestra en la Figura 10, basados en cuando afectan al flujo: *Iniciales*, *Intermedios*, y *Finales*.

Figura 10. Tipos de Eventos

- **Actividad:** En la Figura 11 se muestra una actividad que se representa con un rectángulo redondeado y es un término genérico para el trabajo que hace una compañía. Una actividad puede ser atómica o compuesta. Los tipos que hay son: *Tareas* y *Sub-Procesos.* El Sub-Proceso se distingue por una pequeña marca de suma en la parte central inferior de la figura.

Figura 11. Tarea

- **Gateway (compuerta):** una *gateway* se representa por la típica figura de diamante (Figura 12) y se usa para controlar la divergencia o convergencia de la secuencia de flujo. Así, esto determina las tradicionales decisiones, así como la creación de nuevos caminos, la fusión de estos o la unión. Los marcadores internos indicarán el tipo de control de comportamiento.

Figura 12. Gateway

4.3.1 Objetos conectores

Los objetos de flujo se conectan entre ellos en un diagrama para crear el esqueleto básico de la estructura de un proceso de negocio. Hay tres objetos conectores que hacen esta función. Estos conectores son:

- **Flujo de secuencia:** el flujo de secuencia se representa por una línea sólida como el de la Figura 13 con una cabeza de flecha sólida y se usa para mostrar el orden (la secuencia) en el que las diferentes actividades se ejecutarán en el Proceso.

Figura 13. Flujo de Secuencia

- **Flujo de mensaje:** el flujo de mensaje se representa por un línea discontinua con una punta de flecha hueca como el de la Figura 14 y se usa para mostrar el flujo de mensajes entre dos participantes del proceso separados (entidades de negocio o roles de negocio).

Figura 14. Flujo de mensaje

- **Asociación:** una asociación se representa por una línea de puntos con una punta de flecha de líneas como el de la Figura 15 y se usa para asociar datos, texto, y

otros artefactos con los objetos de flujo. Las asociaciones se usan para mostrar entradas y salidas de las actividades.

Figura 15. Flujo de Asociación

Para los modeladores que requieran o desean más precisión para crear modelos de proceso por motivos de documentación y comunicación, los elementos básicos más los conectores dan la posibilidad de crear fácilmente diagramas comprensible y en el cual podemos ver un ejemplo en la Figura 16.

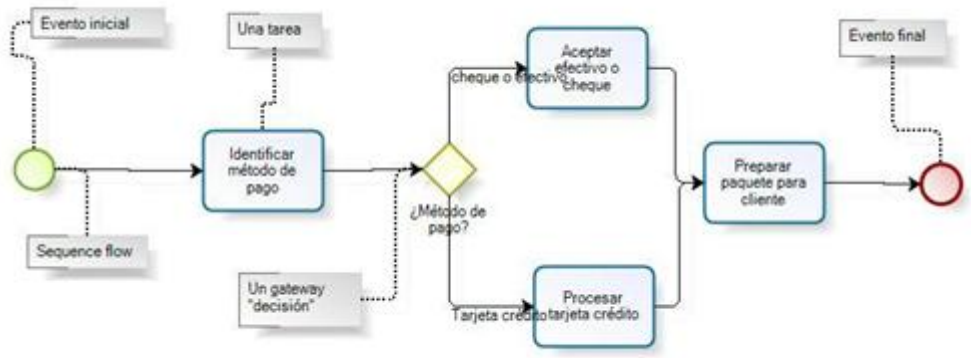


Figura 16. Diagrama proceso simple

Para los diseñadores que necesiten un nivel más alto de precisión, para análisis detallado o que sean manejados por un Business Process Management System (BPMS), existen detalles adicionales que se pueden añadir a los elementos básicos. Un ejemplo lo vemos en la Figura 17.

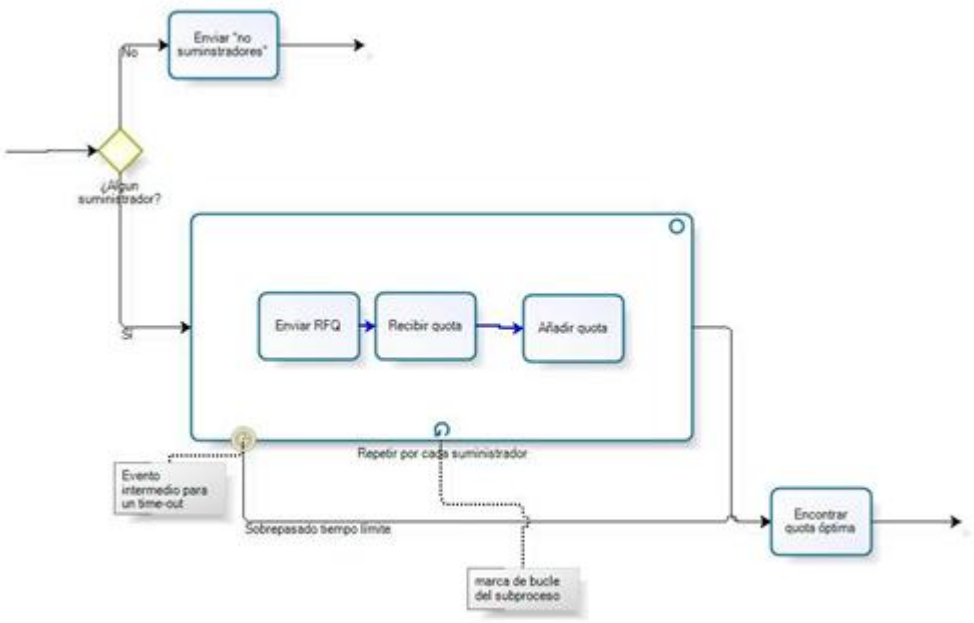


Figura 17. Diagrama proceso detalle

4.3.2 Swimlanes (canales)

Muchas metodologías de modelado de procesos usan el concepto de *swimlanes* como un mecanismo para organizar actividades en categorías separadas visualmente para ilustrar diferentes capacidades funcionales o responsabilidades. BPMN soporta los swimlanes con dos constructores principales. Los dos tipos de objetos swimlanes (Figura 18) son:

- **Pool:** una *pool* representa un Participante de un Proceso. Además actúa como un contenedor gráfico para particionar un conjunto de actividades desde otros pools, normalmente en el contexto de B2B.
- **Lane:** una *lane* es una sub-partición dentro de un pool y extiende la longitud del pool, verticalmente u horizontalmente. Las lanes se usan para organizar y categorizar actividades.

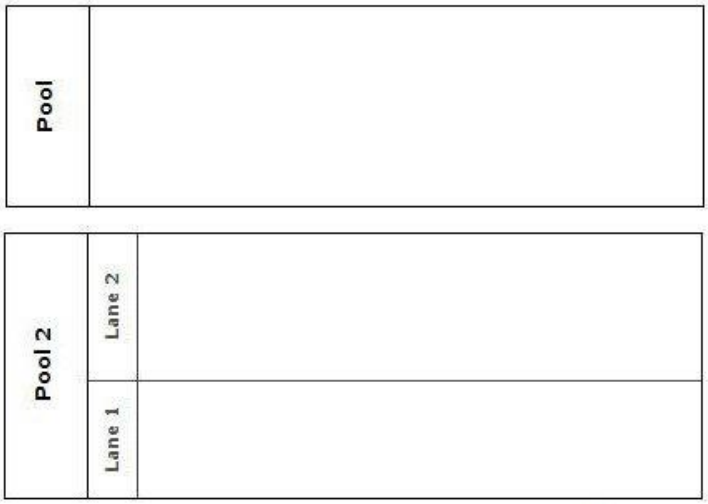


Figura18. Pools y lanes

Las pools se usan cuando un diagrama implica dos entidades de negocio o participantes separados y están físicamente separados en el diagrama. Las actividades dentro de pools separadas se consideran procesos autocontenidos. Así, el flujo de secuencia no debe cruzar el límite de un pool. El flujo de mensajes se define como el mecanismo para mostrar las comunicaciones entre dos participantes, y, de este modo debe conectar dos pools (o los objetos dentro de las pools). Se puede ver un ejemplo completo en la Figura 19.

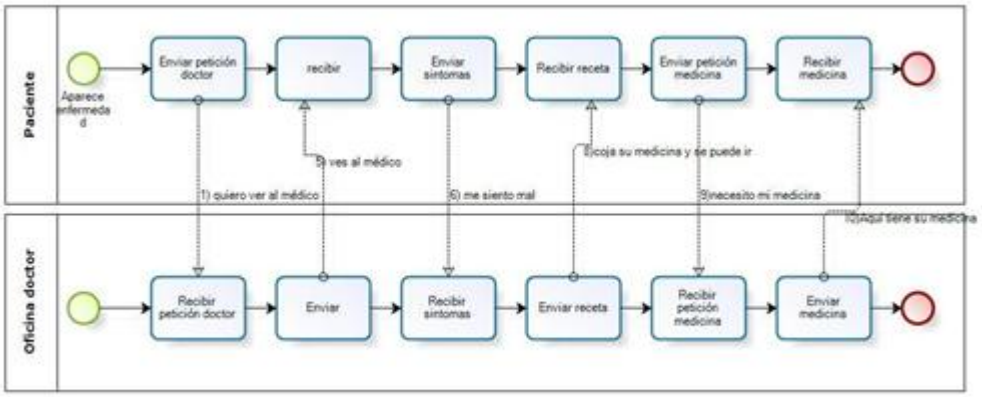


Figura 19. Ejemplo de pools y lanes

Las pistas (lanes) están más estrechamente relacionadas con las metodologías tradicionales de las swimlanes. Las pistas se suelen usar para separar las actividades asociadas con la función o rol de una compañía específica. El flujo de secuencia puede cruzar los límites de las pistas dentro de un pool, pero el flujo de mensajes no puede ser

usado entre objetos de flujo en pistas de mismo pool. Un ejemplo de pistas asociadas a roles en la Figura 20.

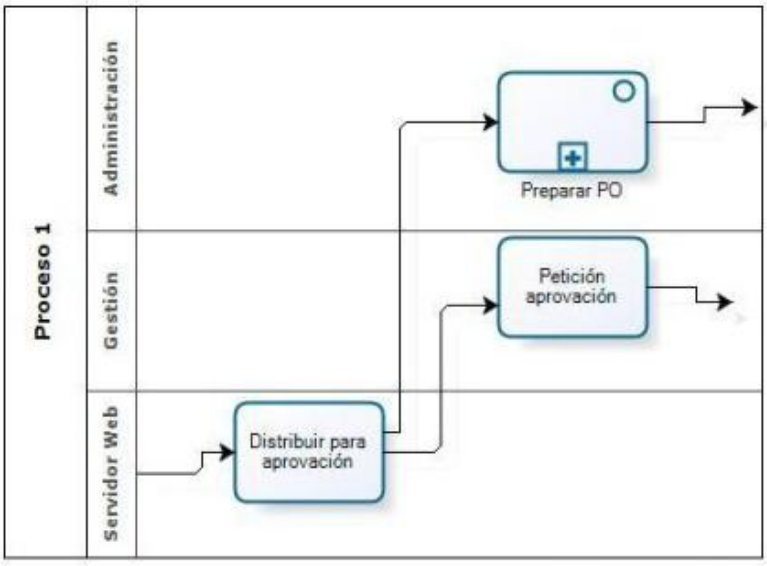


Figura 20. Diagrama proceso

4.3.3 Artefactos

BPMN fue diseñado para permitir a los modeladores y las herramientas de modelado un poco de flexibilidad a la hora de extender la notación básica y a la hora de habilitar un contexto apropiado adicional según una situación específica, como para un mercado vertical (por ejemplo, seguros o banca). Se puede añadir cualquier número de artefactos a un diagrama como sea apropiado para un contexto de proceso de negocio específico. La versión actual de la especificación de BPMN sólo tiene tres tipos de artefactos BPD predefinidos, los cuales son:

- **Objeto de datos:** los objetos de datos (Figura 21) son un mecanismo para mostrar como los datos son requeridos o producidos por las actividades. Están conectados a las actividades a través de asociaciones

Figura 21. Objeto de datos

- **Grupo:** un grupo (Figura 22) es representado por un rectángulo redondeado con línea discontinua. El agrupamiento se puede usar documentación o análisis, pero no afecta al flujo de secuencia.

Figura 22. Grupo

Anotación: las anotaciones (Figura 23) son mecanismos para que un modelador pueda dar información textual adicional.

Figura 23. Anotación

Los modeladores pueden crear sus propios tipos de artefactos, que añaden más detalle (Figura 24) sobre cómo se ejecuta el proceso – bastante a menudo para mostrar las entradas y las salidas de las actividades del Proceso. Sin embargo, la estructura básica del proceso, determinada por las actividades, gateways, y flujos de secuencia, no se cambia por añadir artefactos al diagrama.

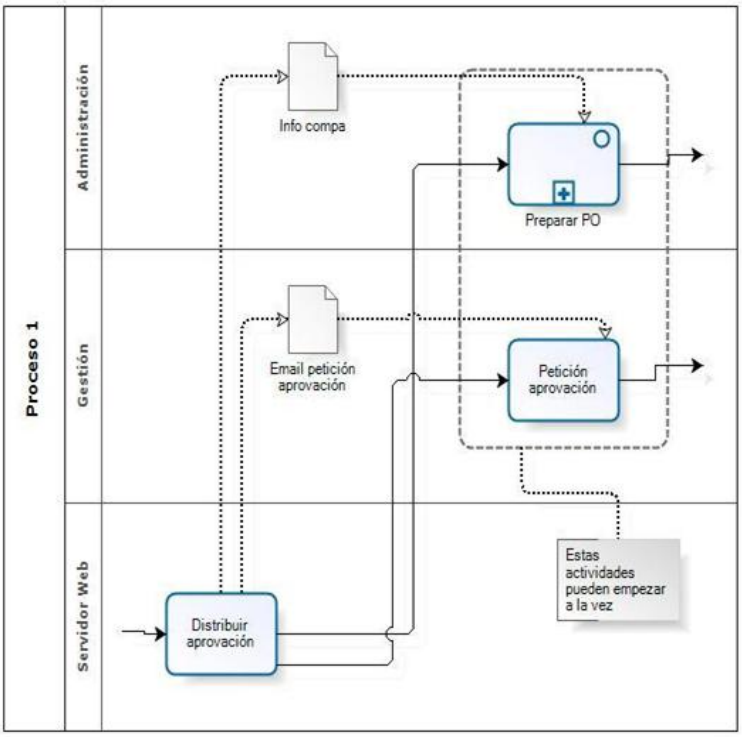


Figura 24. Diagrama ejemplo

4.4 Uso General de BPMN

El modelado de procesos de negocio se usa para comunicar una amplia variedad de información a diferentes audiencias. BPMN está diseñado para cubrir muchos tipos de modelados y para permitir la creación de segmentos de proceso así como procesos de negocio *end-to-end*, con diferentes niveles de fidelidad. Dentro de la variedad de objetivos de modelado de procesos, hay dos tipos de modelos básicos que se pueden crear con un BPD:

- Procesos B2B colaborativos (públicos)
- Procesos de negocio internos (privados)

4.4.1 Procesos B2B colaborativos

Un proceso B2B colaborativo ilustra las interacciones entre dos o más entidades de negocio. Los diagramas para estos tipos de procesos están generalmente desde un punto de vista global. Esto es, no toman la visión de un participante en particular, pero muestra las interacciones entre los participantes. Las interacciones están ilustradas como una secuencia de actividades y los patrones de intercambio de mensajes entre participantes. Las actividades para los participantes son los *"touch-points"* entre

participantes; el proceso define las interacciones que son visibles al público para cada participante. Cuando miramos un proceso en un solo Pool (por ejemplo, para un participante), un proceso público también se llama proceso *abstracto*. Los procesos reales (internos) son como tener más actividades y detalle que lo que se enseña en los procesos B2B colaborativos.

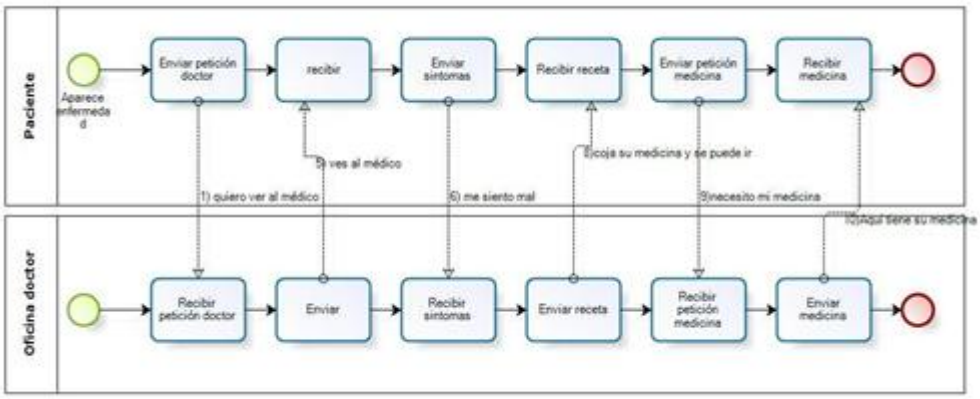


Figura 25. Diagrama con pools y lanes

4.4.2 Procesos de negocio internos

Un proceso de negocio interno se enfocará generalmente en el punto de vista de una única organización de negocio. Aunque los procesos internos suelen mostrar interacciones con participantes externos, definen las actividades que generalmente no están visibles para el público, esto es, privadas.

Si se usan swimlanes entonces un proceso interno estará contenido dentro de un solo Pool. El flujo de secuencia del proceso está por lo tanto contenido dentro de un Pool y no puede cruzar los límites del Pool. El flujo de mensajes puede cruzar los límites del Pool para mostrar las interacciones que existen entre procesos de negocios internos separados. Así, un solo diagrama de procesos de negocio puede mostrar múltiples procesos de negocio privados.

4.5 Diferentes niveles de precisión

El modelado de procesos de negocio suele empezar capturando actividades de alto nivel para luego ir bajando de nivel de detalle dentro de diferentes diagramas. Pueden haber múltiples niveles de diagramas, dependiendo de la metodología usada para desarrollar los modelos. De todas formas, BPMN es independiente de cualquier metodología.

En la Figura 26 tenemos un ejemplo de procesos de alto nivel, capturados para un caso de estudio de BPMN. Se trata de una serie de sub procesos con tres puntos de decisión.

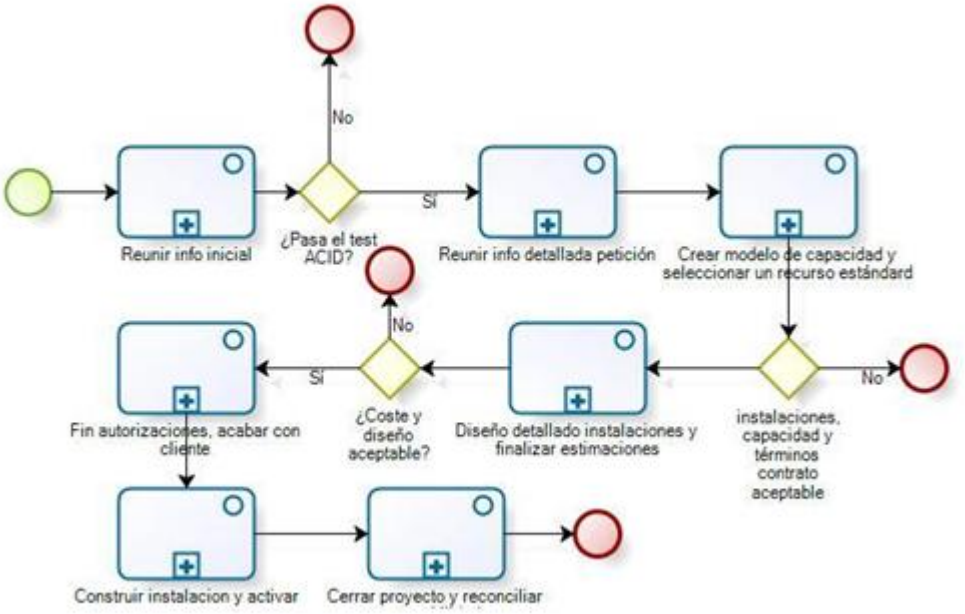


Figura 26. Diagrama de procesos a alto nivel

A continuación, en la Figura 27 se baja de nivel para mostrar en detalle el primer sub proceso: dos pools, una para los clientes y otra para la compañía suministradora. Este diagrama muestra un proceso de negocio interno para la compañía y un proceso abstracto para el cliente. Las actividades de la compañía están particionadas con pistas o *lanes* para mostrar los roles/departamentos responsables de su rendimiento.

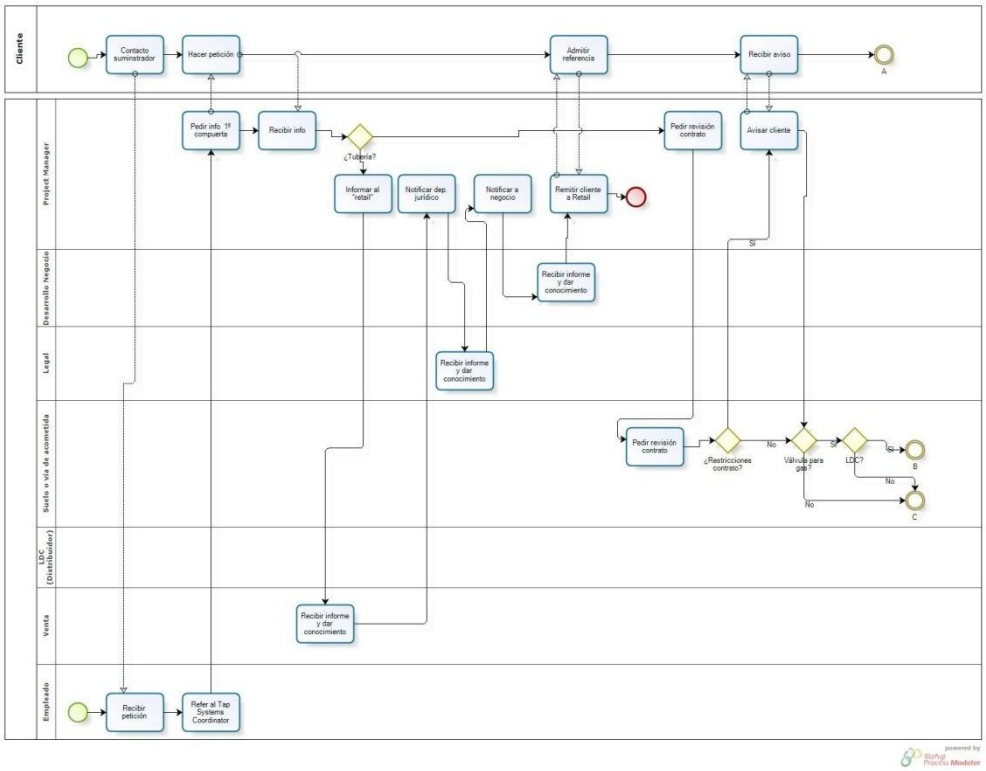


Figura 27. Diagrama proceso a bajo nivel

4.6 Valor de modelar en BPMN

Los miembros de BPMI Notation Working Group representan un gran segmento de la comunidad de modelado de procesos de negocio y han llegado a un consenso y presentan BPMN como la notación de modelado de procesos de negocio estándar. El desarrollo de BPMN es un paso importante para reducir la fragmentación que existe con la gran cantidad de herramientas de modelado de procesos y notaciones. El BPMI Notation Working Group porta una gran experiencia con muchas de las notaciones existentes y trabajan para consolidar las mejores ideas de todas estas notaciones para crear una sola notación estándar. Una única notación bien definida reduce la confusión entre los usuarios IT y de negocios. Ejemplos de otras notaciones o metodologías que fueron revisadas son: diagramas de actividades de UML, UML EDOC Business Processes, IDEF, ebXML BPSS, Diagrama de flujo de actividades-decisiones (ADF), RosettaNet, LOVeM, Cadenas de Eventos-Procesos (EPCs).

Otro factor del desarrollo de BPMN es que, históricamente, los modelos de procesos de negocio desarrollados por la gente de negocios han estado técnicamente separados de las representaciones de procesos requeridas por los sistemas diseñados para implementar y ejecutar dichos procesos. Así, era necesario traducir manualmente los modelos de procesos de negocio originales a los modelos de ejecución. Esas traducciones están sujetas a errores y dificultan a los dueños de los procesos entender la evolución y el rendimiento de los procesos desarrollados.

4.7 Mapear un diagrama BPMN a BPEL4WS

Para ayudar a aliviar el vacío técnico de modelado, un objetivo clave para el desarrollo de BPMN era crear un puente entre la notación de modelado de procesos de negocios y los lenguajes de ejecución respecto a las Tecnologías de la Información que implementan los procesos que hay dentro de un sistema. Los objetos gráficos de BPMN, más un buen número de atributos de estos objetos, se han mapeado al Business Process Execution Language para Web Services (BPEL4WS v1.1), el estándar de facto para la ejecución de procesos. En la Figura 28 tenemos un segmento de un proceso de negocio que marca el mapeo con BPEL4WS.

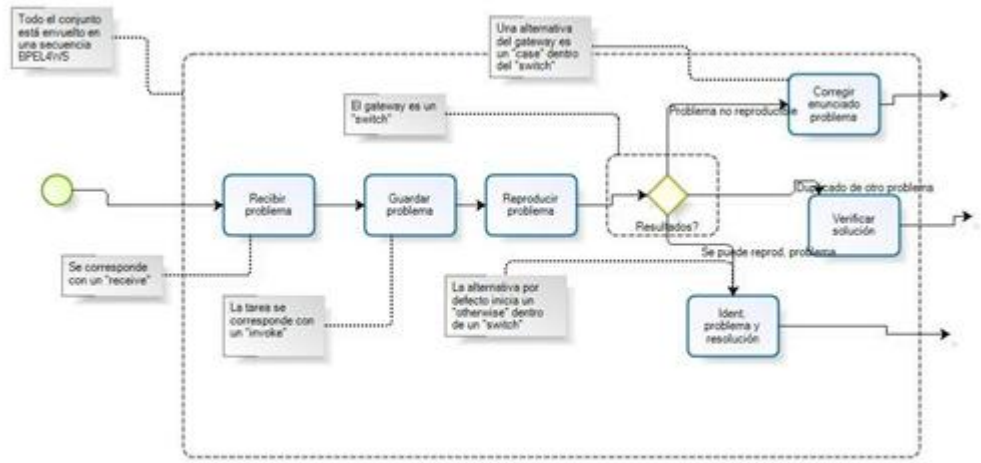


Figura 28. Diagrama ejemplo

4.8 El futuro de BPMN

Aunque la especificación de BPMN se encuentra actualmente en su versión estable 1.2 (existe una versión 2.0 en fase beta), muchas compañías se han comprometido a soportar e implementar dicha especificación. El futuro inmediato dará un punto de experiencia entre usuarios y vendedores que permitirá, mediante feedback, afinar detalles de la especificación, en concreto con BPEL4WS. En las siguientes versiones de mantenimiento es de esperar un esfuerzo en estandarización de los artefactos para que soporten modelado de negocios generales y dominios de negocios verticales (seguros, manufacturación, finanzas). Además, se está intentando encajar BPMN en un mayor contexto de modelado de negocios de alto nivel (incluyendo reglas de negocio y estrategias de negocio).

4.9 Notación BPMN

Aunque la versión actual es la 1.2, a continuación se va a dar una breve descripción de la notación existente en la versión 1.1 de la notación BPMN [5] que es la más utilizada en estos momentos.

4.9.1 Compuertas

Compuerta Exclusiva Basada en Datos: Las compuertas exclusivas basadas en datos (Figura 29) son los tipos de compuerta más usados comúnmente. El conjunto de salidas para decisiones exclusivas basadas en datos está basado en la expresión booleana contenida en el atributo expresión de condición del flujo de secuencia que sale de la compuerta. Estas expresiones usan los valores de los datos del proceso para determinar qué camino debe tomar (por eso el nombre "basada en datos").

Figura 29. Compuerta Exclusiva Basada en Datos

Compuerta Exclusiva Basada en Eventos: Esta decisión (Figura 30) representa un punto de bifurcación en el proceso donde las alternativas están basadas en eventos que ocurren en ese punto del proceso, más que en la evaluación de expresiones usando datos del proceso. Un evento específico, usualmente la recepción de un mensaje, determina cuál de los caminos debe ser tomado. Por ejemplo, si una compañía está esperando por una respuesta de un cliente, ejecutará un conjunto de actividades si el cliente responde "Si" y otro conjunto de actividades si el cliente responde "No". La respuesta del cliente determina que camino es tomado. La identidad del mensaje determina que camino se ha tomado. Esto es, si el mensaje "Si" y el mensaje "No" son diferentes – no son el mismo mensaje con diferentes valores dentro de una propiedad del mensaje. La recepción del mensaje puede ser modelada con una tarea de tipo Recepción o un Evento Intermedio con un mensaje. Además de los mensajes, también se podrían usar temporizadores.

Figura 30. Compuerta Exclusiva Basada en Eventos

Compuerta Paralela: Las compuertas paralelas (Figura 31) proveen un mecanismo para crear y sincronizar flujos paralelos. Estas compuertas no son requeridas para crear un flujo paralelo, pero pueden ser usadas para clarificar el comportamiento de situaciones complejas donde un conjunto de compuertas son usadas y un flujo paralelo es requerido.

Figura 31. Compuerta Paralela

Compuerta Inclusiva: Esta decisión (Figura 32) representa un punto de bifurcación donde las alternativas están basadas en expresiones condicionales contenidas dentro de los Flujos de Secuencia de salida. Sin embargo, en este caso, la evaluación verdadera de una expresión condicional no excluye la evaluación de otras expresiones condicionales. Todos los Flujos de Secuencia con cuya evaluación sea verdadera, serán recorridos por un token. De algún modo es como un agrupamiento de decisiones binarias (Si/No) relacionadas – y pueden ser modeladas como tal. Ya que cada camino es independiente, todas las combinaciones de caminos pueden ser tomadas, desde ninguna hasta todas. Sin embargo, debe ser diseñado para que al menos un camino sea tomado.

Figura 32. Compuerta Inclusiva

Compuerta Compleja: BPMN incluye una compuerta compleja (Figura 33) para manejar situaciones que no son fácilmente manejadas usando otros tipos de compuertas. Las compuertas complejas también pueden ser usadas para combinar un conjunto de compuertas simples enlazadas en una situación simple y más compacta. Los

modeladores pueden proveer expresiones complejas que determinen el comportamiento de unión y/o división de la compuerta.

Figura 33. Compuerta Compleja

4.9.2 Actividades

Múltiples Instancias (Figura 34) de la misma actividad que se empiezan en paralelo o secuencialmente, por ejemplo, para cada elemento de un *line* en un orden.

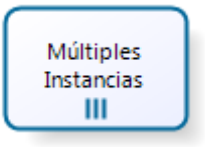


Figura 34. Múltiples Instancias

Actividad Bucle (Figura 35) se itera si la condición se evalúa a cierto. La condición es testeada tanto antes como después de la ejecución de la actividad.

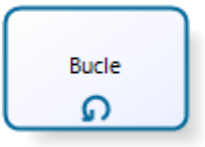


Figura 35. Actividad Bucle

Flujo de Secuencia: define el orden de ejecución de las actividades (Figura 36).

Figura 36. Flujo de Secuencia

Flujo Condicional: tiene una condición asignada que define si el flujo se usará o no (Figura 37).

Figura 37. Flujo Condicional

Flujo por defecto: es la rama que se elige por defecto si todas las otras condiciones se evalúan a falso.

Figura 38. Flujo por defecto

Tarea: Es una actividad atómica (Figura 39) que es incluida dentro de un Proceso. Una Tarea es usada cuando el trabajo en el Proceso no es descompuesto. Generalmente, un usuario final y/o una aplicación son los encargados de ejecutar la Tarea.

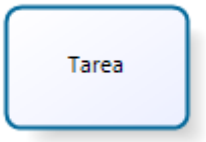


Figura 39. Tarea

Subproceso: es una actividad que se descompone (Figura 40). Se puede contraer para ocultar los detalles.

Figura 40. Subproceso

Subproceso Expandido: contiene un diagrama BPMN válido (Figura 41).

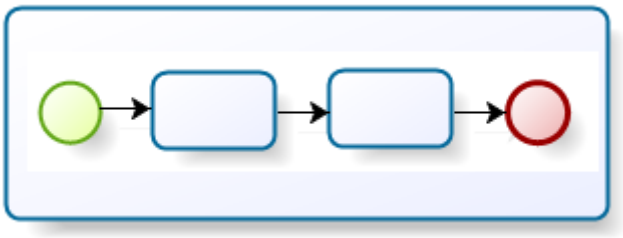


Figura 41. Subproceso Expandido

4.9.3 Datos

Objeto de Datos: Los Objetos de Datos (Figura 42) proveen información acerca de cómo los documentos, datos y otros objetos son usados y actualizados durante el Proceso. Aunque el nombre "Objeto de Datos" puede implicar un documento electrónico, puede usarse para representar diversos tipos de objetos, tantos electrónicos como físicos.

Figura 42. Objeto de Datos

Adjuntando un objeto de datos con una **Asociación Indirecta** (Figura 43) a un flujo de secuencia indica una transferencia de información entre las actividades involucradas.

Figura 43. Asociación Indirecta

Una **Asociación Directa** (Figura 44) indica un flujo de información. Un objeto de datos puede ser leído al empezar una actividad o escrito una vez completado.

Figura 44. Asociación Directa

Una **Asociación Bidireccional** (Figura 45) indica que los datos del objeto está siendo modificado, por ejemplo, lectura y escritura durante la ejecución de una actividad.

Figura 45. Asociación Bidireccional

4.9.4 Transacciones

Una **Transacción** (Figura 46) es un conjunto de actividades que pertenecen juntas y que se ejecutan de manera lógica, es decir, si se ejecuta una, deben ejecutarse todas o cancelarse todas.

Figura 46. Transacción

Adjuntando **Eventos de Cancelación Intermedios** (Figura 47) se indica las reacciones de cancelación de una transacción. Actividades dentro de una transacción son compensadas por la cancelación.

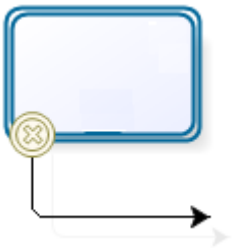


Figura 47. Evento de Cancelación Intermedio

4.9.5 Documentación

Un conjunto arbitrario de objetos pueden ser definidos como un **Grupo** (Figura 48) para mostrar que van de la mano de una manera lógica.

Figura 48. Grupo

Cualquier objeto puede asociarse con una **Anotación de Texto** (Figura 49) para ofrecer una documentación adicional.

Figura 49. Anotación de Texto

4.9.6 Swimlanes

Pools y **Lanes** (Figura 50) representan responsabilidades para actividades en un proceso. Un Pool o Lane puede ser una organización, un rol o un sistema. Lanes sub-dividen Pools u otras Lanes de manera jerárquica.

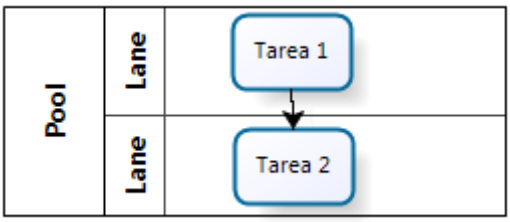


Figura 50. Pools y Lanes

Pools Colapsadas (Figura 51) ocultan todos los detalles de los procesos contenidos.

Figura 51. Pool Colapsada

Los **flujos de mensajes** (Figura 52) simbolizan información que fluye a través de los límites de la organización. Los flujos de mensajes pueden ser adjuntados a pools, actividades o eventos de mensajes.

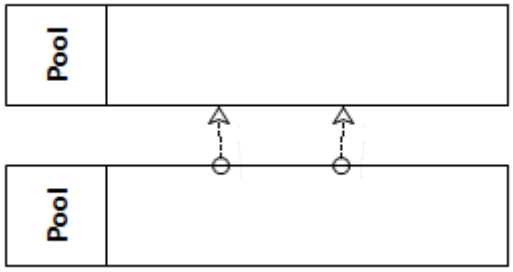


Figura 52. Flujo de mensajes

El orden del intercambio de mensajes (Figura 53) puede ser especificado combinando flujos de mensaje y flujos de secuencia.

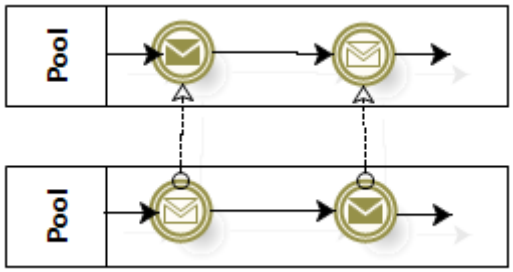
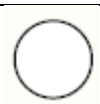
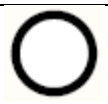


Figura 53. Orden en el intercambio de mensajes

4.9.7 Tipología de Eventos

	Comienzo	Intermedios		Finales	Descripción
	Recoger		Lanzar		
Plano					Eventos sin tipo, normalmente indican donde empieza o termina el proceso.

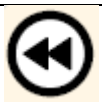
Mensaje					Recibir y enviar mensajes.
Temporizador					Eventos de temporizador cíclicos, puntos en el tiempo o timeouts.
Error					Atrapa o lanza errores definidos.
Cancelación					Reacción a las operaciones canceladas o disparo de anulaciones.
Compensación					Manejador de compensaciones o de triggers.
Condicional					Reacción al cambio de condiciones de negocio o integración de reglas de negocio.
Señal					Señalización a través de diferentes procesos. Una señal lanzada puede ser atrapada por múltiples tiempos.
Múltiples					Capturar o lanzar un evento de una serie de eventos.
Enlace					Conectar dos secciones de un proceso.
Terminación					Disparando la terminación de un proceso.

Tabla 2. Tipología de los eventos disponibles en BPMN

Capítulo 5. Tecnologías de Workflows Actuales

En este capítulo se van a presentar aquellas las tecnologías más usadas en workflows.

5.1 Windows Workflow Foundation

5.1.1 Introducción

Windows Workflow Foundation (WWF) [51] es el modelo de programación, motor y herramientas para generar aplicaciones basadas en flujos de trabajo en Windows. Está compuesto por un espacio de nombres (Windows.Workflow), un motor de flujo de trabajo en proceso y diseñadores para Visual Studio. Windows Workflow Foundation es un marco que permite a los usuarios crear flujos de trabajo de sistema o humanos en sus aplicaciones para Windows. Se puede utilizar para resolver escenarios simples como mostrar los UI de controles básicos en datos proporcionados por el usuario o los escenarios complejos que se producen en las empresas grandes, como procesamiento del orden y control de inventario.

Posibles escenarios que se incluyen en Windows Workflow Foundation:

- Habilitar el flujo de trabajo dentro de las aplicaciones empresariales.
- Flujos de página del interfaz del usuario.
- Flujos de trabajo centrado en el documento.
- Flujos de trabajo humanos.
- Flujos de trabajo compuesto por aplicaciones orientadas a servicios.
- Flujos de trabajo controlados por las reglas empresariales.
- Flujos de trabajo para la administración de los sistemas.

5.1.2 Tipos de Flujos de Trabajo soportados

WWF soporta dos tipos de flujos de trabajo, tal y como se describe en [2]:

- Flujo Secuencial: Es aquel donde se ejecutan una serie de actividades en una secuencia predefinida. Esta secuencia puede incluir operaciones de control de flujo similares a los que han existido durante años en lenguajes de programación. Por ejemplo, operadores de decisión (if…else) o de iteración (while). Se utiliza para flujos donde nos interesa que el control lo tenga primordialmente el proceso definido.

- Flujo de Máquina de Estados: En este flujo de actividades se ejecutan dependiendo del estado en el que se encuentre una máquina de estados, así como de las transiciones entre estos estados (que pueden ser iniciadas por las actividades mismas). Se utiliza normalmente para flujos donde se prefiere que el control recaiga mayormente en los usuarios del sistema.

En la tabla 3 se muestra una tabla con las principales diferencias entre ambos tipos de flujos de trabajo.

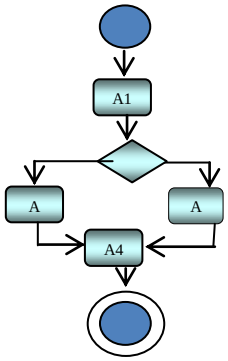
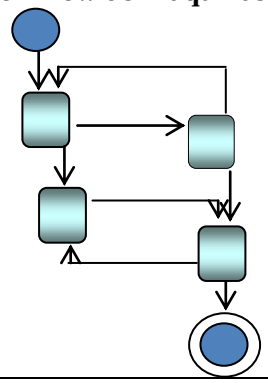
Proceso de negocio	**Modelo de Workflow**
Procesos de control de flujo • Secuencias bien entendidas • Lo más natural es la secuencialidad • El proceso es la guía • Conveniente para la automatización de procesos	**Workflow Secuencial** 
Procesos guiados por eventos • Guiado por eventos externos • Impredecible secuencia de eventos • Muchas alternativas de negocio • Salto a cualquier actividad	**Workflow de Máquinas de Estado** 

Tabla 3. Comparativa entre los diferentes tipos de flujos de trabajo

5.1.3 Ejemplo de Secuenciación

Un paradigma de flujo de trabajo típico comienza con una situación de inicialización, que en este caso es una rama de decisiones (IfElseBranchActivity). En el ejemplo de la Figura 54, se ve un ejemplo relativamente simple del flujo de trabajo que sigue un proceso de aprobación de un documento a ser publicado. La condición inicial determina si el documento que se trata ha sido denegado para la publicación (Refusal) o si sigue el proceso de publicación (Editorial_Work / Publication_Queue / Publication) y finalmente, la finalización del flujo de trabajo es la notificación al autor en ambos casos.

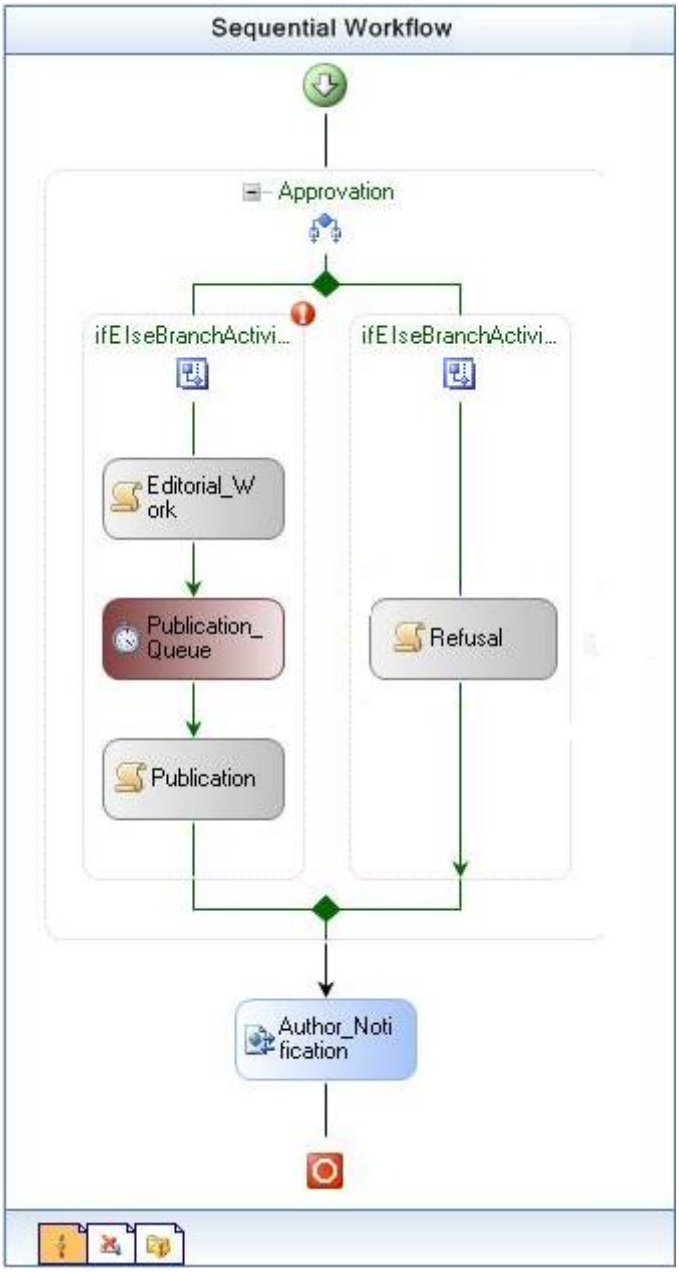


Figura 54 Ejemplo workflow en WWF

5.1.4 Ejemplo de puesta en marcha

5.1.4.1 Creación de un proyecto

Cuando instalamos las Visual Studio Extensions para WWF se agregan varios templates al VS para ayudarnos a crear nuestros proyectos de workflow. En este caso crearemos un **Sequential Workflow Console Application** como muestra en la Figura 55.

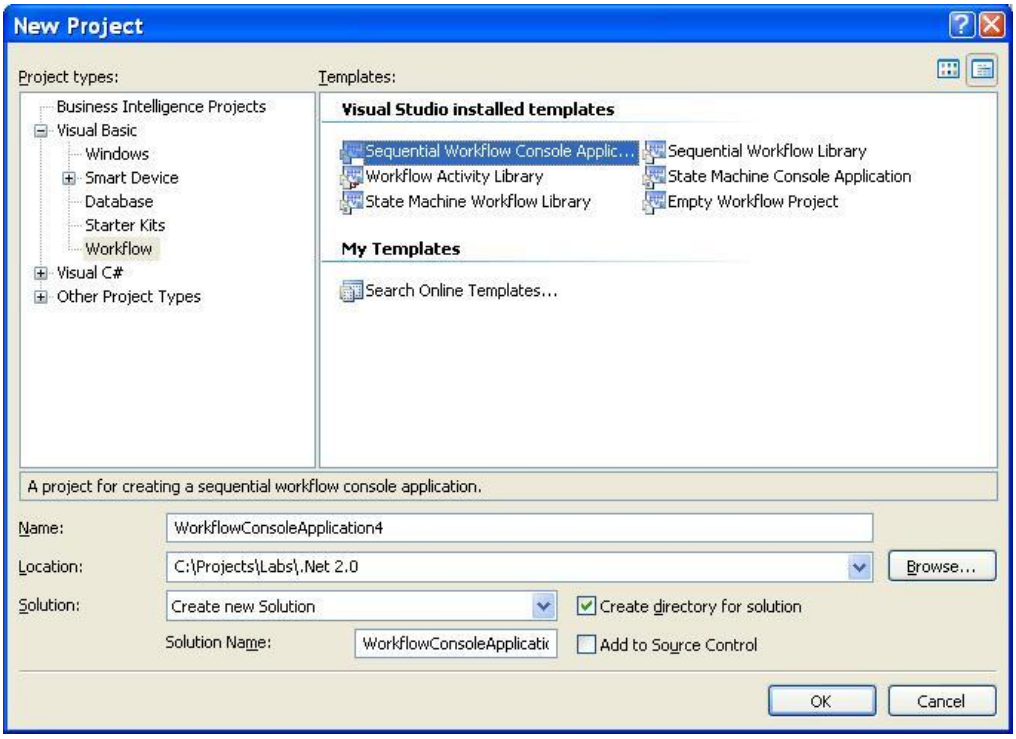


Figura 55. Nuevo proyecto

Esto crea por defecto un proyecto con un archivo Module1.vb que contiene el código de inicialización del workflow:

```
Imports System
Imports System.Collections.Generic
Imports System.Text
Imports System.Threading
Imports System.Workflow.Runtime
Imports System.Workflow.Runtime.Hosting


Module Module1
   Class Program
      Shared WaitHandle As New AutoResetEvent(False)

      Shared Sub Main()
         Dim workflowRuntime As New WorkflowRuntime()
         workflowRuntime.StartRuntime()

         AddHandler workflowRuntime.WorkflowCompleted, AddressOf OnWorkflowCompleted
```

```
        Dim type As System.Type = GetType(Workflow1)
        workflowRuntime.StartWorkflow(type)

        WaitHandle.WaitOne()

        workflowRuntime.StopRuntime()
    End Sub


  Shared Sub OnWorkflowCompleted(ByVal sender As Object, ByVal e As WorkflowCompletedEventArgs)
        WaitHandle.Set()
    End Sub

  End Class
End Module
```

Crea además un archivo Workflow1.vb que contiene la definición del workflow cuya declaración de clase es la siguiente:

```
Partial Public Class Workflow1
  Inherits SequentialWorkflow
```

Esta clase está ligada a la ejecución del workflow, cada actividad que agreguemos en el diseñador, eventos que manejemos y demás tendrán su correspondiente código aquí.

5.1.4.2 Diseño del flujo

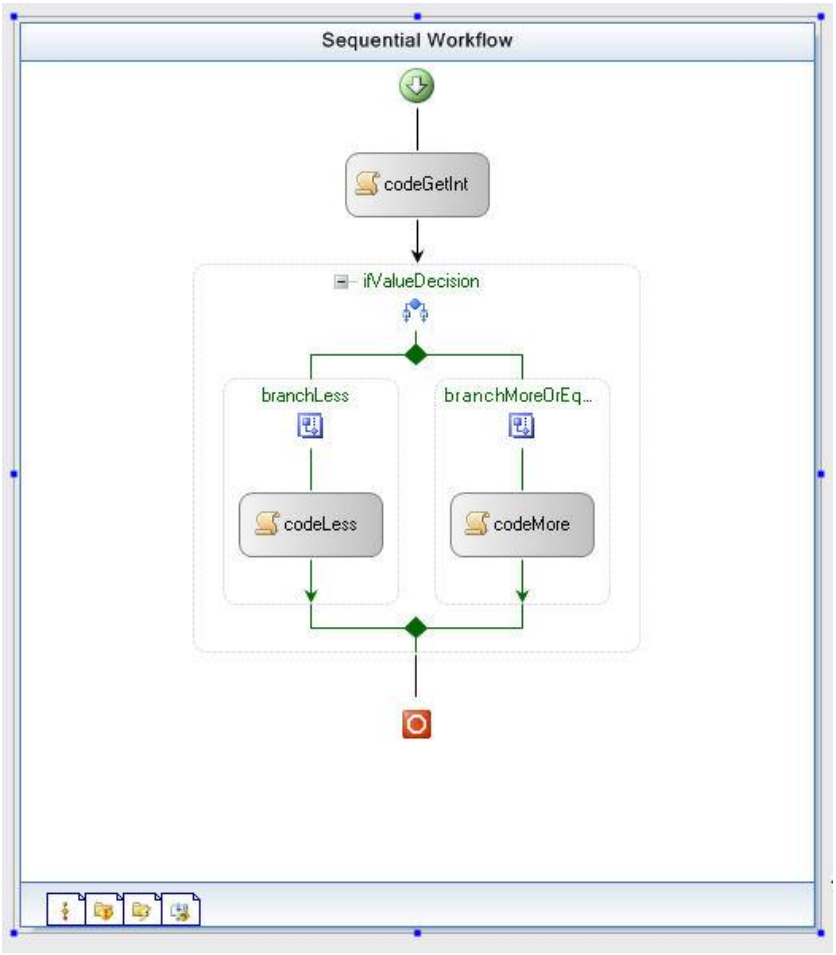


Figura 56. Diseño del flujo

El workflow de la Figura 56 utiliza dos tipos de actividades: "Code" e "IfElse". La actividad del tipo Code está ligada a la ejecución directa de código. Veamos el ejemplo de la actividad codeGetInt.

Esta actividad tiene en la propiedad ExecuteCode el procedimiento que se va a ejecutar cuando el flujo llegue a la misma, que en este caso es "codeGetInt_ExecuteCode":

```
Private Sub codeGetInt_ExecuteCode(ByVal sender As System.Object, ByVal e As System.EventArgs)
    DecisionValue = Integer.Parse(Console.ReadLine())
End Sub
```

El procedimiento lee una línea de la consola, la convierte en un entero y guarda el valor en una variable a nivel de clase denominada "DecisionValue".

La siguiente instrucción en una del tipo IfElse denominada ifValueDecision que tiene dos caminos posibles (pueden agregarse más). El primer camino posee una condición de

evaluación que en caso de ser verdadera (Figura 57) hace que el flujo alcance las actividades que se encuentran en esa rama. En caso de ser falsa se procesan las actividades del brazo alternativo.

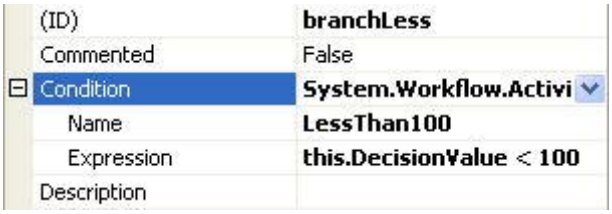


Figura 57. Propiedades del branch de la izquierda

Cada uno de los brazos posee una actividad de tipo Code que escribe un mensaje en la consola:

```
Private Sub codeLess_ExecuteCode(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Console.ForegroundColor = ConsoleColor.Green
    Console.WriteLine("The number is lower than 100")
End Sub


Private Sub codeMore_ExecuteCode(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Console.ForegroundColor = ConsoleColor.Blue
    Console.WriteLine("The number is equal or greater than 100")
End Sub
```

Luego termina el flujo.

Adicional a las ejecuciones de las actividades se sobrescribieron dos eventos del workflow, el de inicialización y de terminación:

```
Private Sub Workflow1_Initialized(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Console.WriteLine("Write a number")
End Sub

Private Sub Workflow1_Completed(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Console.ResetColor()
    Console.WriteLine("Press ENTER to End")
    Console.ReadLine()
End Sub
```

5.1.4.3 Flujo para excepciones

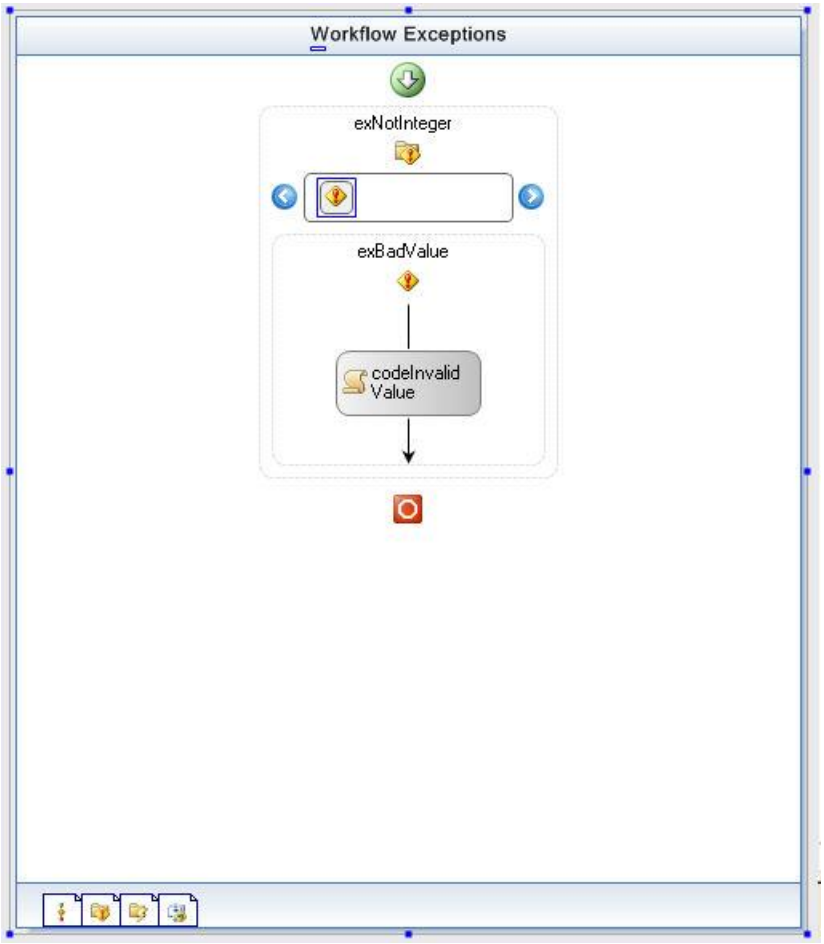


Figura 58. Flujo para excepciones

Es posible definir un flujo para controlar las excepciones que se producen durante la ejecución. En el presente caso, como vemos en la Figura 58 se agregó una actividad "ExcepcionHandler" para que capture las excepciones del tipo "System.FormatExcepcion" que es el error en el cual se incurriría si intentamos parsear como entero algo que no lo es. Cuando el Integer.Parse falla y produce una excepción la ejecución pasa a la actividad exBadValue y luego a la actividad codeInvalidValue que procesa el siguiente código:

```
  Private Sub codeInvalidValue_ExecuteCode(ByVal sender As System.Object, ByVal e AsSystem.EventAr
gs)
      Console.ForegroundColor = ConsoleColor.Red
      Console.WriteLine("You must type an integer")
      Console.ResetColor()
      Console.WriteLine("Press ENTER to End")
      Console.ReadLine()
  End Sub
```

Luego termina la ejecución.

5.1.4.4 Test

Por último, veamos ahora que pasa cuando ejecutamos el ejemplo:

Figura 59. Flujo correcto

Figura 60. Flujo incorrecto

5.2 jBPM

5.2.1 Introducción

jBPM (del inglés, java Business Process Management) [15] es un motor de WF implementado en lenguaje Java que funciona sobre servidores JBoss sobre la plataforma J2EE. jBPM es una solución open source para proveer de un conjunto de herramientas para la gestión e interpretación de WF, incluye además de una definición formal basada en BPEL, una herramienta gráfica de diseño de WF que permite fácilmente la integración de procesos de negocio arrastrando y soltando primitivas de WF.

jBPM es un sistema extensible de administración de flujo de trabajo. jBPM cuenta con un lenguaje de proceso para expresar gráficamente procesos de negocio en términos de tareas, estados de espera para expresar gráficamente procesos de negocio en términos de tareas, estados de espera para comunicación asíncrona, temporizadores, acciones automatizadas, etc. Para unir estas operaciones jBPM cuenta con un mecanismo de control de flujo.

5.2.2 Características

jBPM se puede utilizar con la misma simpleza que una biblioteca java. Pero también puede utilizarse en ambientes donde es esencial contar con un alto nivel de producción mediante la implementación en un servidor de aplicaciones J2EE en clúster.

jBPM se puede configurar con cualquier base de datos y se puede implementar en cualquier servidor de aplicación.

El flujo de trabajo central y la funcionalidad BPM tienen un formato de biblioteca java. Esta biblioteca incluye un servicio para almacenar, actualizar y recuperar información de proceso de la base de datos jBPM.

El servidor de aplicación jBoss pre-configurado tiene instalados los siguientes componentes:

- El componente central jBPM, organizado como archivo de servicio.
- Una base de datos integrada con las tablas jBPM.
- La aplicación web de la consola jBPM puede ser utilizado por participantes del proceso así como por administradores jBPM.

- El programador jBPM para la ejecución de los temporizadores. El programador está configurado en el kit de inicio en forma de servlet. El servlet va a dar origen a una secuencia para monitorizar y ejecutar los temporizadores.
- El ejecutor de comandos del jBPM para la ejecución asíncrona de comandos. El ejecutor de comandos también se configura como servlet. El servlet da origen a una secuencia para monitorizar y ejecutar los comandos.
- Un proceso de muestra ya está implementado en la base de datos jBPM.

5.2.3 Diseñador gráfico de proceso jBPM

jBPM también incluye una herramienta gráfica de diseño [8]. El diseñador es una herramienta gráfica para crear los procesos de negocio en el cual vemos en la Figura 61 un ejemplo. El diseñador gráfico de proceso es un plugin eclipse.

La característica más importante de la herramienta gráfica de diseño es que incluye soporte tanto para las tareas del analista de negocios como para el desarrollador técnico. Esto permite una transacción armónica desde el modelado de procesos de negocio a la implementación práctica.

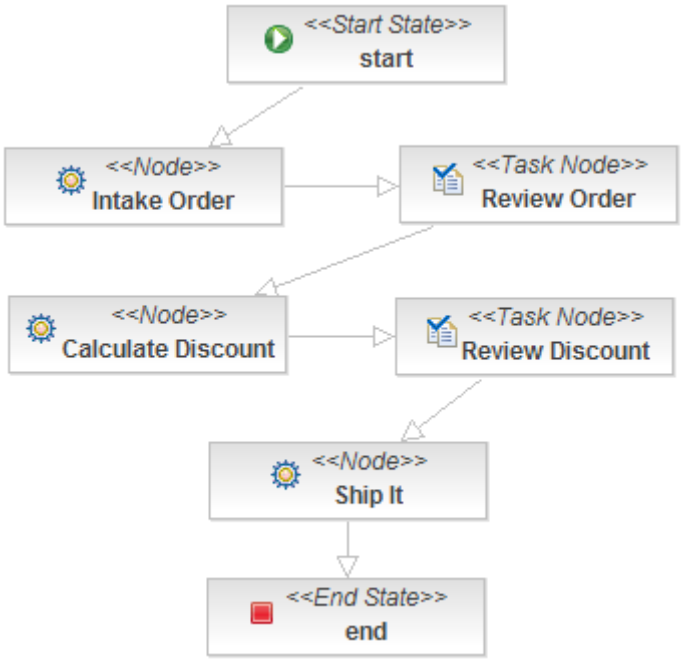


Figura 61. Diagrama generado con jBPM

El editor gráfico permite crear una definición de proceso visual. Los nodos y las transiciones entre los nodos se pueden añadir, modificar o eliminar. La definición del

proceso se guarda como un documento XML que puede ser almacenado en un sistema de archivos y enviado a una instancia de jBPM (base de datos). Cada vez que se implementa la instancia de proceso jBPM se versionará y mantendrá almacenadas las copias anteriores. Esto permite que los procesos que están en marcha puedan seguir utilizando la instancia de proceso anterior que ellos empezaron. Las nuevas instancias de proceso usarán la última versión de la definición del proceso.

5.2.4 Ejemplo de puesta en marcha

Lo primero es iniciar Eclipse e iniciar el proyecto (Figura 62).

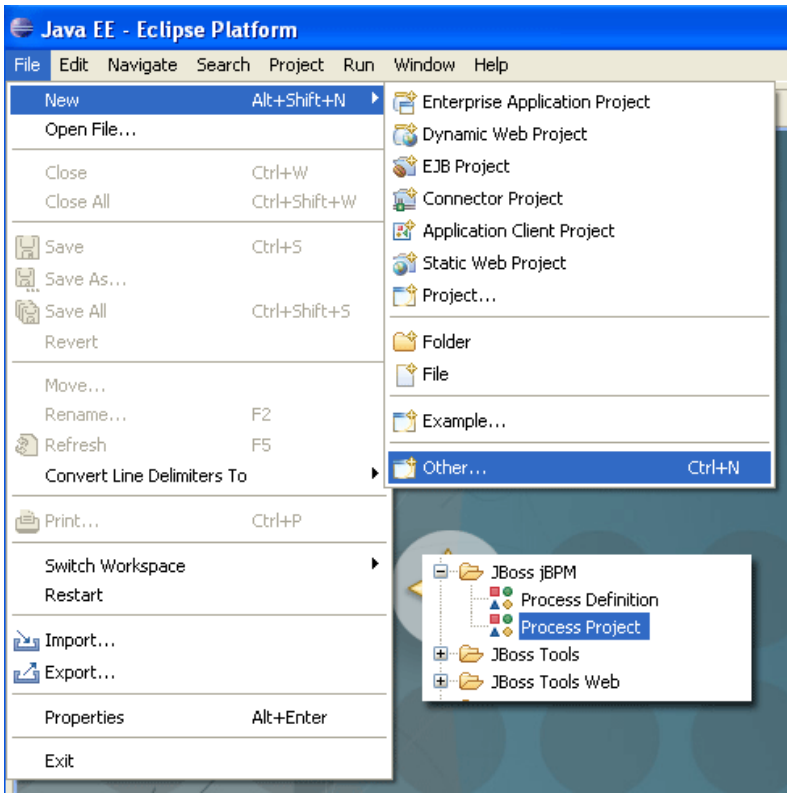


Figura 62. Nuevo proyecto

Elegimos Process Project y procedemos a darle un nombre (Figura 63).

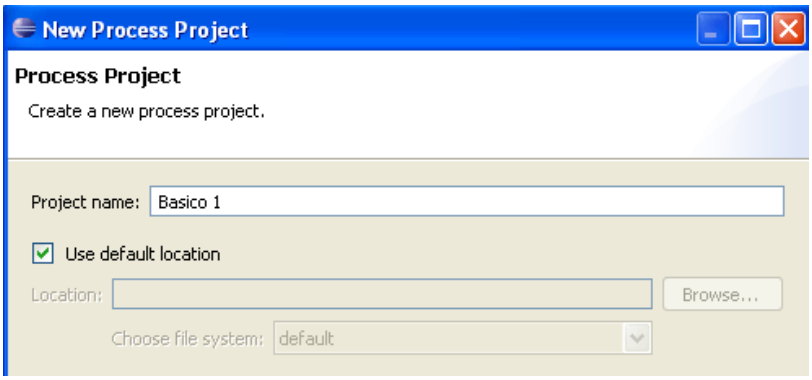


Figura 63. Nombre del nuevo proyecto

Nos pide configurar el JBoss jBPM Runtime, así que buscamos el directorio donde nos hayamos instalado el jBPM (Figura 64 y 65).

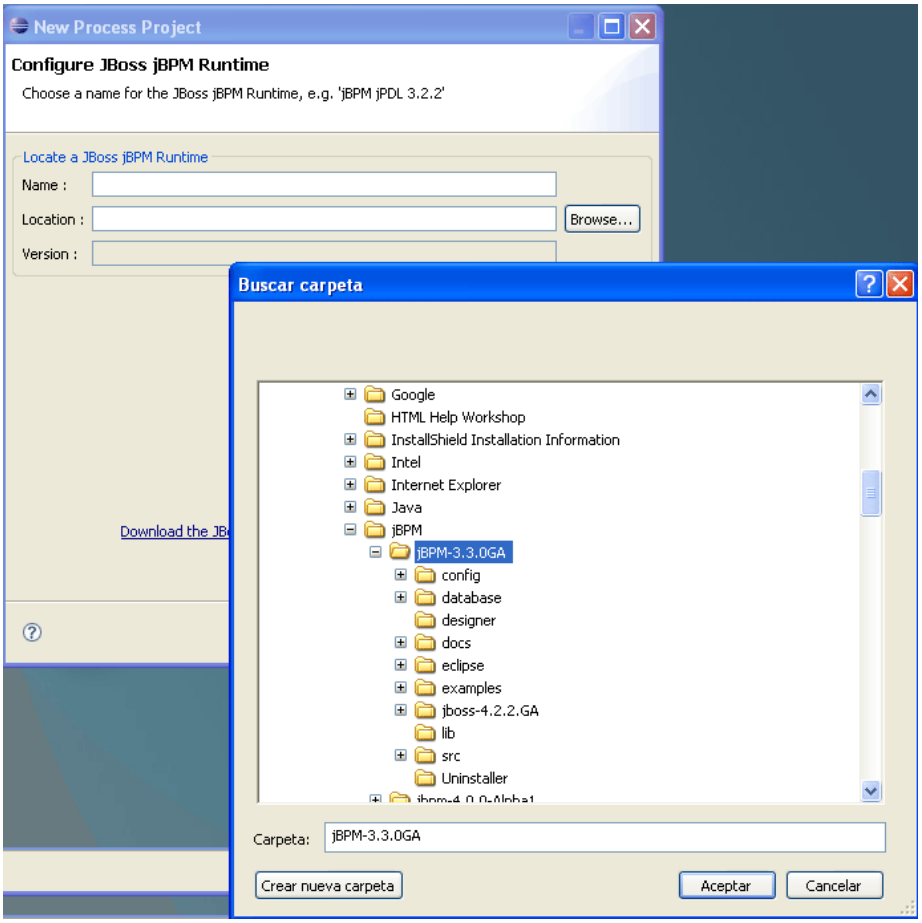


Figura 64. Configurar JBoss

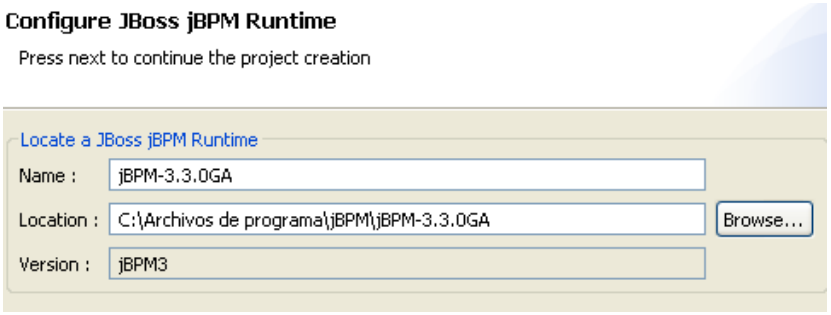


Figura 65. Configurar JBoss

Una vez añadido el nombre, se elige el Core jBPM, como en la Figura 66 al que habéis dado nombre en el paso anterior y seleccionáis el cuadro para que te genere el simple process definition, action handler y JUnit test. Generalmente es más fácil empezar con esto aunque para el ejemplo básico voy a crear el proyecto totalmente vacío.

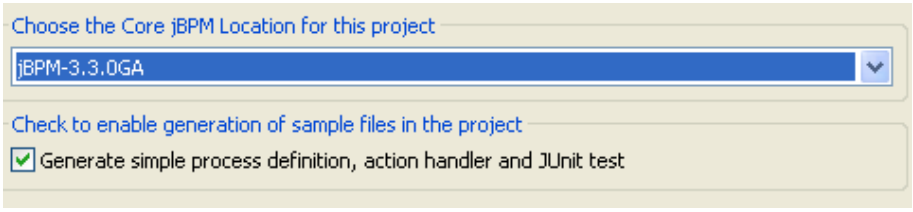


Figura 66. Elegir el Core jBPM

Le dais a finalizar y ya tenéis la estructura necesaria para empezar, mostrada en la Figura 67.

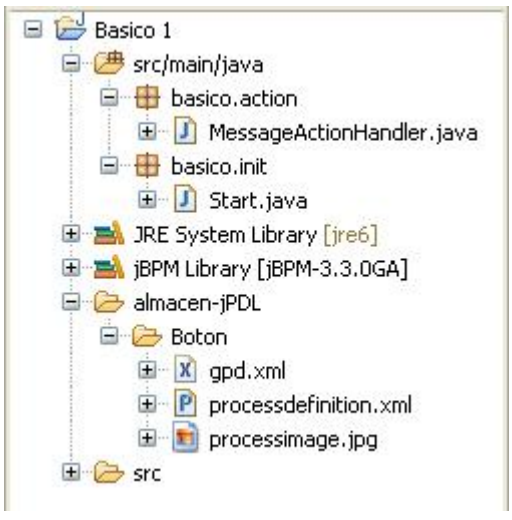


Figura 67. Estructura del proyecto

El paquete basico.init es donde se va a iniciar la aplicación, un paquete basico.action donde está el handle de las acciones que definamos en los workflows y un directorio almacen-jPDL donde se van a almacenar los workflows que se vayan creando.

Para crear un nuevo workflow, se hace botón derecho sobre el directorio donde se quiere crear y elegir New → Other → Process definition (Figura 68).

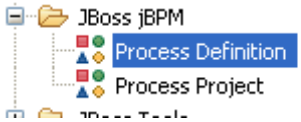


Figura 68. Crear un nuevo workflow

El workflow es muy simple, simplemente simulará el encendido de un botón y finalizará el workflow (Figura 69).

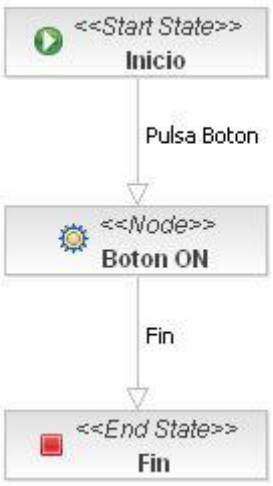


Figura 69. Workflow de ejemplo

Después de dibujarlo, hay que añadir las acciones. Una manera de hacerlo es mediante código modificando directamente el xml. El contenido es el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>

<process-definition  xmlns="urn:jbpm.org:jpdl-3.3"  name="Boton">
<start-state name="Inicio">
<transition to="Boton ON" name="Pulsa Boton">
<action name="action" class="basico.action.MessageActionHandler">
<message>Pulsando Boton</message>
</action>
</transition>
</start-state>
<node name="Boton ON">
<action name="action" class="basico.action.MessageActionHandler">
<message>Boton en posicion ON</message>
</action>
<transition to="Fin" name="Fin">
<action name="action" class="basico.action.MessageActionHandler">
<message>Proceso terminado</message>
</action>
</transition>
</node>
```

```
<end-state name="Fin"></end-state>
</process-definition>
```

La aplicación donde cargaremos y ejecutaremos el workflow es el siguiente:

```
package basico.init;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import org.jbpm.graph.def.ProcessDefinition;
import org.jbpm.graph.exe.ProcessInstance;
public class Start {

public static void main(String[] args) throws FileNotFoundException {
FileInputStream fis = new FileInputStream("almacen-jPDL/Boton/processdefinition.xml");
ProcessDefinition processDefinition = ProcessDefinition.parseXmlInputStream(fis);
ProcessInstance instance = new ProcessInstance(processDefinition);
instance.signal("Pulsa Boton");
instance.signal("Fin");
}
}
```

Simplemente cargamos el fichero processdefinition.xml, parseamos el xml en la estructura ProcessDefinition y creamos una instancia del workflow. Con esto ya tendremos nuestro workflow preparado para recibir las órdenes.

La primera es "Pulsa Botón" que nos llevará al nodo Botón ON y ejecutará una acción, definida de la siguiente manera:

```
package basico.action;

import org.jbpm.graph.def.ActionHandler;
import org.jbpm.graph.exe.ExecutionContext;
public class MessageActionHandler implements ActionHandler {

private static final long serialVersionUID = 1L;

String message;

public void execute(ExecutionContext context) throws Exception {
context.getContextInstance().setVariable("message", message);
System.out.println(message);
}
}
```

El resultado final una vez ejecutado el workflow son los siguientes mensajes en pantalla.

```
Pulsando Botón
Botón en posición ON
Proceso terminado
```

Capítulo 6. La metodología TUNE-UP

6.1 Introducción

Un proyecto de desarrollo o mantenimiento de software tiene por objetivo el conseguir una entrega exitosa del producto. En un proceso iterativo e incremental el trabajo asociado a una entrega del producto se divide en varias versiones (resultantes de cada iteración) siendo la última aquella que coincide con la entrega del producto.

La metodología "ideal" para la planificación y seguimiento de proyectos software debe tener en cuenta ciertas particularidades:

- Resulta más conveniente utilizar un proceso iterativo e incremental
- Gran parte del trabajo se realiza en paralelo
- El trabajo de los agentes es muy colaborativo y dinámico
- La coordinación entre agentes debería estar guiada por workflows
- Es importante conectar la herramienta con el trabajo del agente
- La planificación debe estar conectada con la gestión del producto, es decir, la gestión de requisitos del producto software.

La metodología TUNE-UP [20] y su herramienta de apoyo TUNE-UP Software Process [21], surgen como respuesta a estas particularidades sobre la gestión de proyectos software.

TUNE-UP es una metodología que incorpora aspectos ágiles y tradicionales con un sentido marcadamente pragmático. TUNE-UP se caracteriza fundamentalmente por combinar los siguientes elementos:

- **Modelo iterativo e incremental** para el desarrollo y mantenimiento del software. El trabajo se divide en unidades de trabajo que son asignadas a versiones del producto. Una unidad de trabajo puede ser un nuevo requisito, una mejora o la corrección de un defecto. Se realizan ciclos cortos de desarrollo, entre 3 y 6 semanas, dependiendo del producto.
- **Workflows flexibles** para la coordinación del trabajo asociado a cada unidad de trabajo. Los productos, según sus características, tienen disponibles un conjunto de workflows los cuales se asignan a cada una de las unidades de trabajo. Cada unidad de trabajo sigue el flujo de actividades del workflow para completarla.

Bajo ciertas condiciones se permite saltar hacia adelante o hacia atrás en el workflow, así como cambios de agentes asignados e incluso cambio de workflow. Por ejemplo, las típicas situaciones de re-trabajo en desarrollo de software ocasionadas por detección de defectos se abordan con saltos atrás no explícitos en el workflow. Esta es la característica que ha motivado este trabajo de Tesis y que será detallada en los siguientes capítulos.

- **Proceso de desarrollo dirigido por las pruebas de aceptación (Test-Driven)**. La definición de una unidad de trabajo es básicamente la especificación de sus pruebas de aceptación acordadas con el cliente. A partir de allí, todo el proceso gira en torno a ellas, se estima el esfuerzo de implementar, diseñar y aplicar dichas pruebas, se diseñan e implementan y luego se aplican sobre el producto para garantizar el éxito de la implementación.
- **Planificación y seguimiento continuo centrados en la gestión del tiempo**. En todo momento debe estar actualizado el estado de las versiones, de las unidades de trabajo, y del trabajo asignado a los agentes. El jefe del proyecto puede actuar oportunamente con dicha información, tomando decisiones tales como: redistribuir carga de trabajo entre agentes, cambiar los plazos de la versión, mover unidades de trabajo entre versiones, etc.
- **Control de tiempos**. Los agentes registran el tiempo que dedican a la realización de las actividades, el cual se compara con los tiempos estimados en cada una de ellas, detectando oportunamente desviaciones significativas. Esto permite a los agentes gestionar más efectivamente su tiempo, mejorar sus estimaciones y ofrecer al jefe del proyecto información actualizada del estado de la versión.

TUNE-UP es una metodología que incorpora aspectos de metodologías ágiles y de metodologías tradicionales. Las dos primeras características (proceso iterativo e incremental, y proceso centrado en las pruebas de aceptación) clasifican a TUNE-UP como metodología ágil, sin embargo, las otras características están más próximas de lo que sería una metodología tradicional.

Este trabajo se centra precisamente en las dos últimas características que veremos en capítulos posteriores.

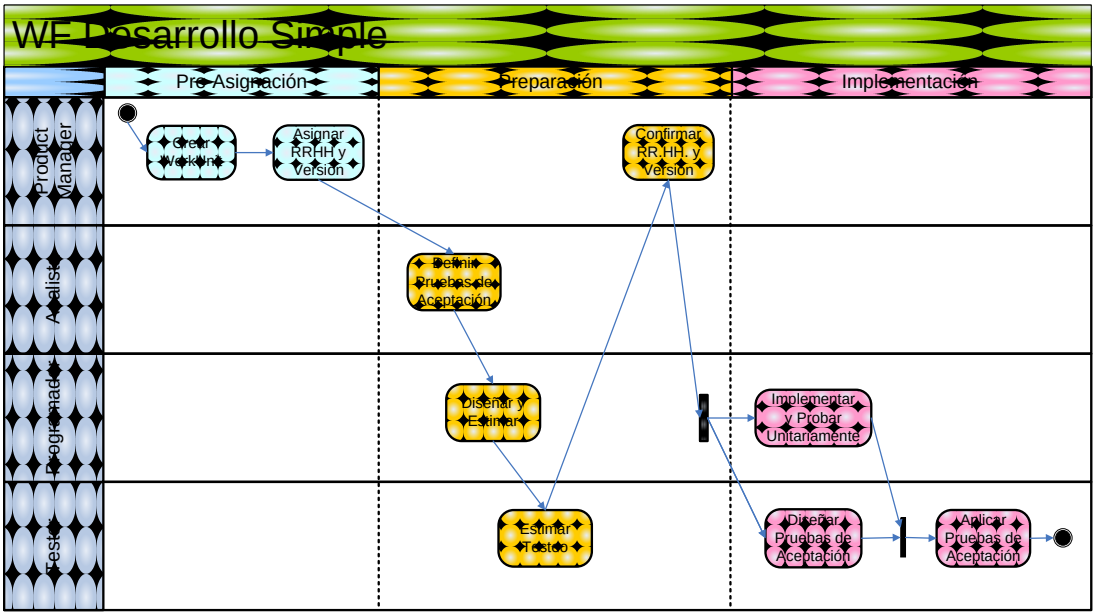


Figura 70. Un workflow de desarrollo simple para unidades de trabajo

La Figura 70 ilustra un workflow básico para el desarrollo de una unidad de trabajo. Un workflow, en general, incluye actividades asociadas a tres fases del proceso:

- **Pre-Asignación:** actividades realizadas hasta la asignación de los RRHH y versión. La unidad de trabajo ha sido identificada pero aún no tiene una prioridad como para asignarle los RRHH y versión.
- **Preparación:** se pueden realizar cuando la unidad de trabajo ha alcanzado cierta prioridad y se le han asignado los RRHH y una versión. Se comienza a trabajar en la preparación de la unidad de trabajo, y debería concluirse antes del inicio de la versión objetivo en la cual se implementará. Incluye el análisis, las revisiones y estimaciones para su implementación.
- **Implementación:** se realizan durante la versión objetivo. Incluye la implementación, aplicación de pruebas e implantación.

Las actividades de cada workflow pueden variar significativamente dependiendo de factores tales como: cantidad y especialización de agentes participantes, validaciones o negociaciones predeterminadas con el cliente, características del producto (necesidad de migración, traducción, etc.), niveles y actividades de pruebas (unitarias, de integración, de aceptación, pruebas de regresión, automatización de pruebas), etc. Cada unidad de trabajo en una iteración del proyecto podría tener su propio workflow. Sin embargo, en la práctica basta con disponer de un reducido conjunto de workflows que permitan cubrir los tipos de unidades de trabajo que se presentan en el proyecto.

Desde el punto de vista del agente responsable, el estado en el que se puede encontrar una unidad de trabajo asociada a la actividad que debe realizar puede ser:

- **Por Llegar:** el agente está asignado a la actividad pero aún no ha recibido la unidad de trabajo pues está en alguna actividad anterior en el workflow.
- **Pendiente:** el agente ha recibido la unidad de trabajo en la actividad pero aún no ha comenzado a trabajar en ella.
- **Activa:** el agente está trabajando en la actividad (y el sistema está registrando tiempo dedicado a ella).
- **Pausada:** el agente ha interrumpido su trabajo en la actividad (y se ha detenido el registro de tiempo.
- **Finalizada:** el agente ha terminado la actividad (la unidad de trabajo ha pasado automáticamente a las actividades siguientes en el workflow asociado).
- **Omitida:** la actividad ha sido omitida, es decir, se ha decidido no realizar la actividad (se ha saltado hacia adelante en el workflow). A menos que la unidad de trabajo dé un salto hacia atrás, la unidad de trabajo no pasará por la actividad.

La Figura 71 muestra los cambios de estados posibles de una unidad de trabajo en una actividad.

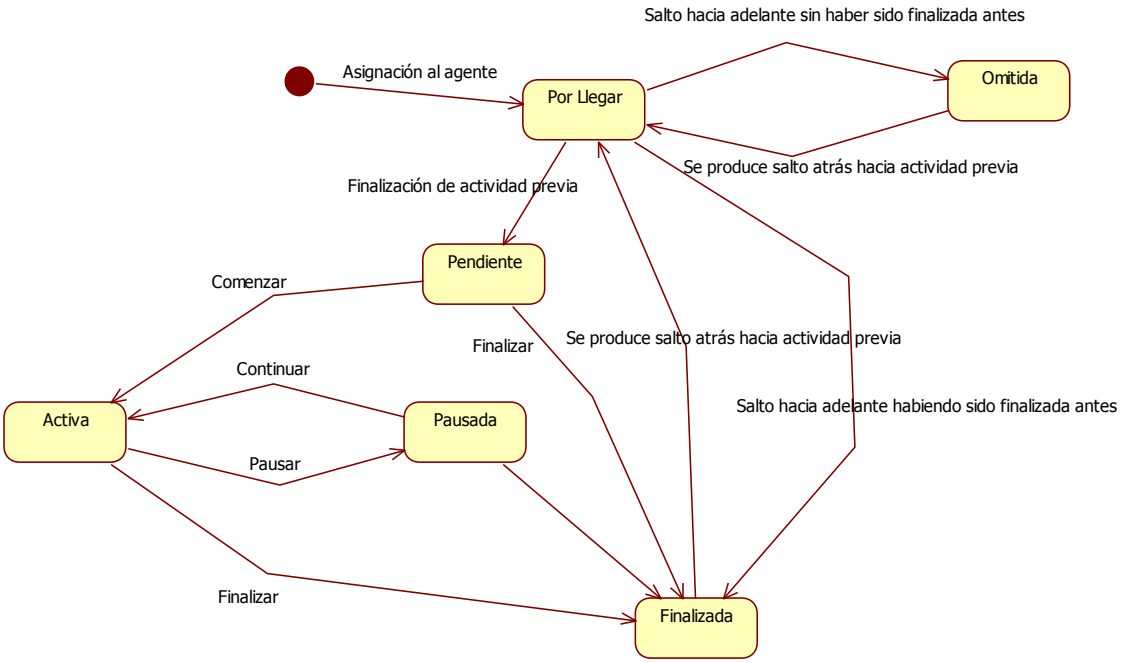


Figura 71. Posibles estados de una unidad de trabajo en una actividad

6.2 TUNE-UP Software Process

TUNE-UP Software Process es una herramienta de apoyo para la aplicación efectiva de la metodología TUNE-UP. La herramienta está formada por tres módulos principales:

- Planificador Personal
- Gestor de Unidades de Trabajo
- Planificador de Versiones.

6.2.1 Planificador personal

El Planificador Personal (PEP) es la herramienta que utilizará un agente para gestionar sus trabajos asignados durante su jornada laboral. La herramienta le ayudará a seleccionar qué actividad realizar en base a las prioridades y llevar un control sobre las tiempos dedicados y restantes de cada actividad. En la Figura 72 podemos ver el listado de actividades a la izquierda y sus unidades de trabajo correspondientes a la derecha.

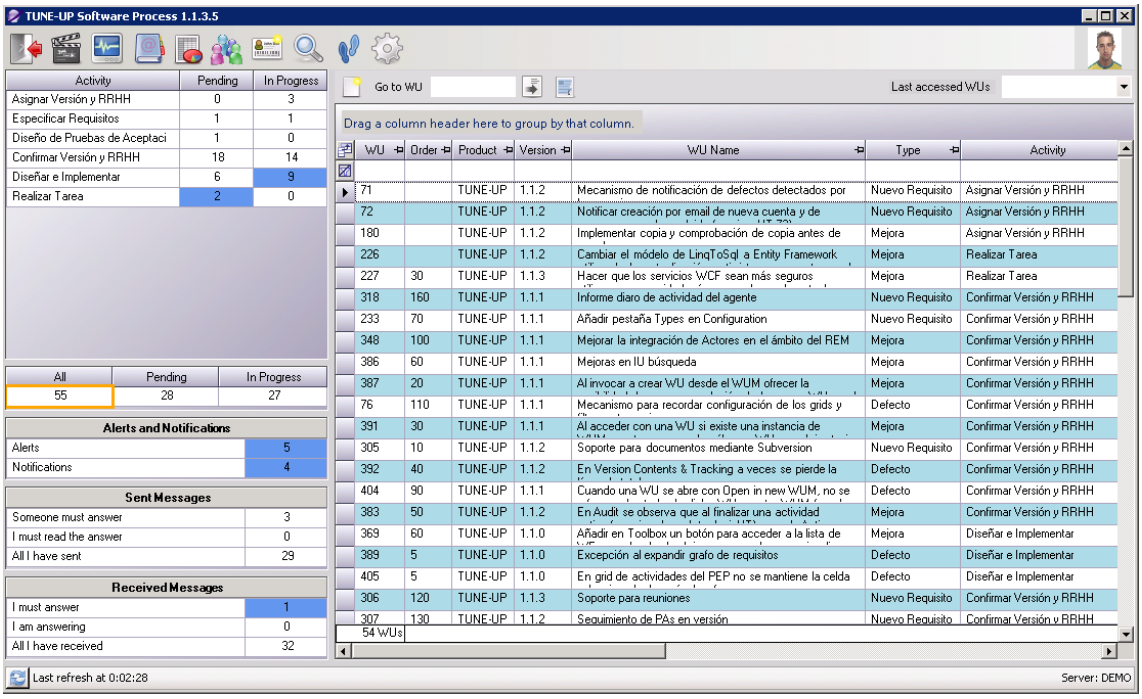


Figura 72. Fragmento de interfaz del Planificador Personal

6.2.2 Gestor de Unidades de Trabajo (WUM)

El agente para empezar a trabajar con una unidad de trabajo debe acceder mediante el GUT, el cual le sirve de apoyo para realizar su tarea en múltiples aspectos:

- Gestión de seguimiento: la herramienta ofrece datos de seguimiento, a partir de los que es posible saber por qué actividades ha pasado la unidad de trabajo,

quienes han trabajado en ella, en qué estado está y a su vez permite modificar el estado de la misma como se puede ver en la Figura 73.

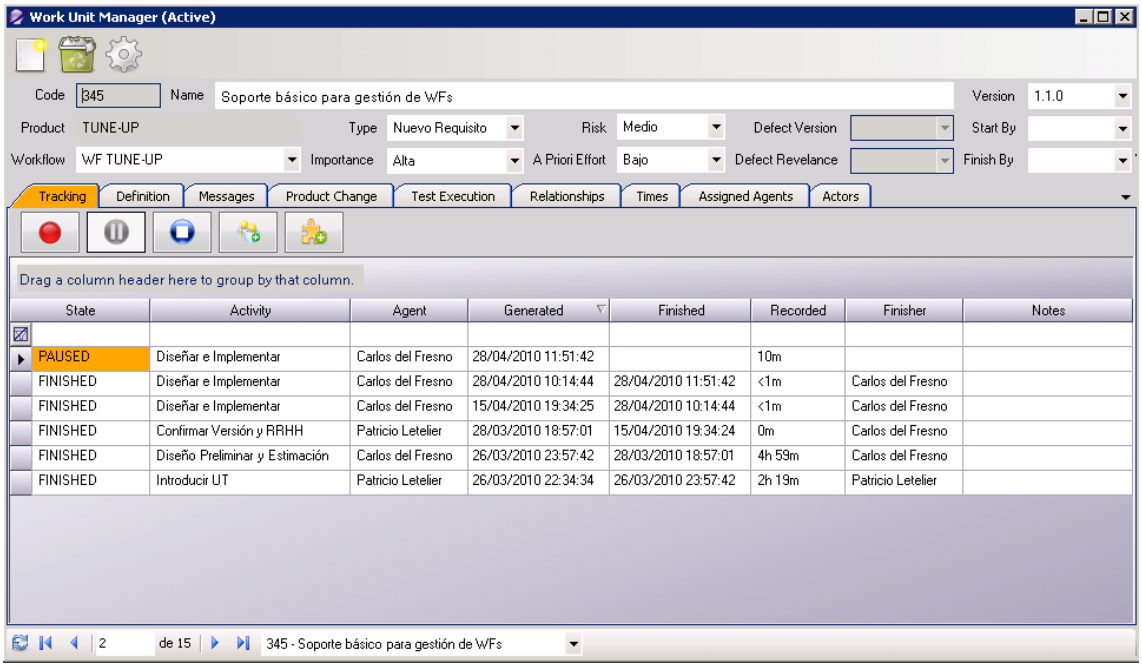


Figura 73. Gestor de Unidades de Trabajo – Pestaña Seguimiento

- Gestión de tiempos: el agente puede llevar un control del tiempo consumido en la unidad de trabajo y de sus estimaciones como se puede ver en la Figura 74.

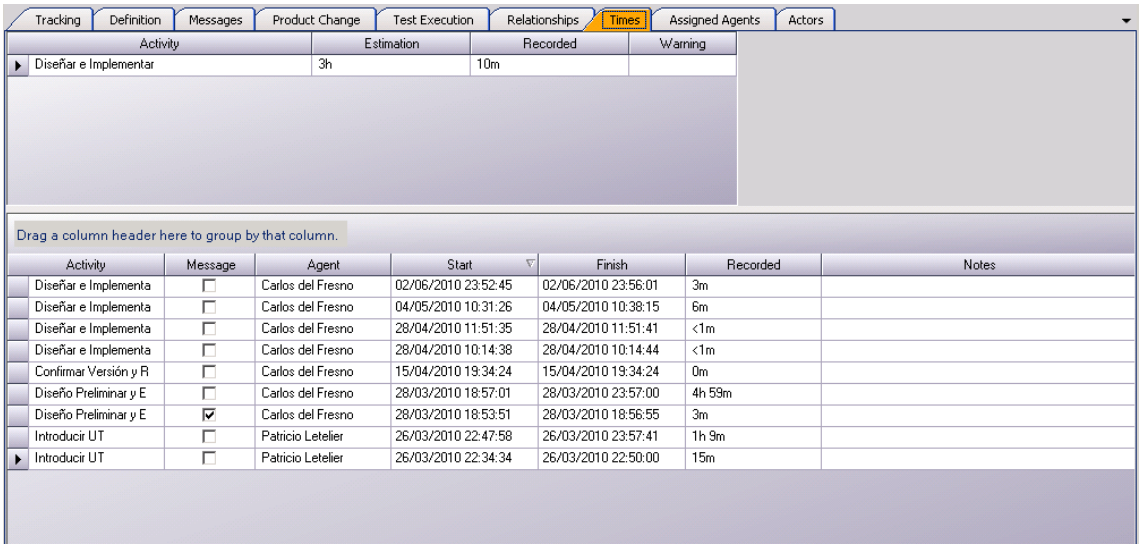


Figura 74. Gestor de Unidades de Trabajo – Pestaña Tiempos

- Definición: dentro de una unidad de trabajo es posible dar una descripción textual más completa de aquello que se va a realizar.
- Gestión de mensajes: mecanismo sencillo de comunicación entre los agentes para comentar respecto de la unidad de trabajo.

- Gestión de relaciones: relaciones de dependencia de la unidad de trabajo con respecto de otras
- Gestión de asignaciones: asignación de agentes a actividades de la unidad de trabajo y asignación a una versión.

6.2.3 Version Content & Tracking (VCT)

El VCT es una herramienta que ayuda a gestionar los productos, sus versiones, los workflows disponibles para cada producto, los agentes por defecto asignados a las actividades de los workflows y realizar el seguimiento continuo del estado actual de las versiones.

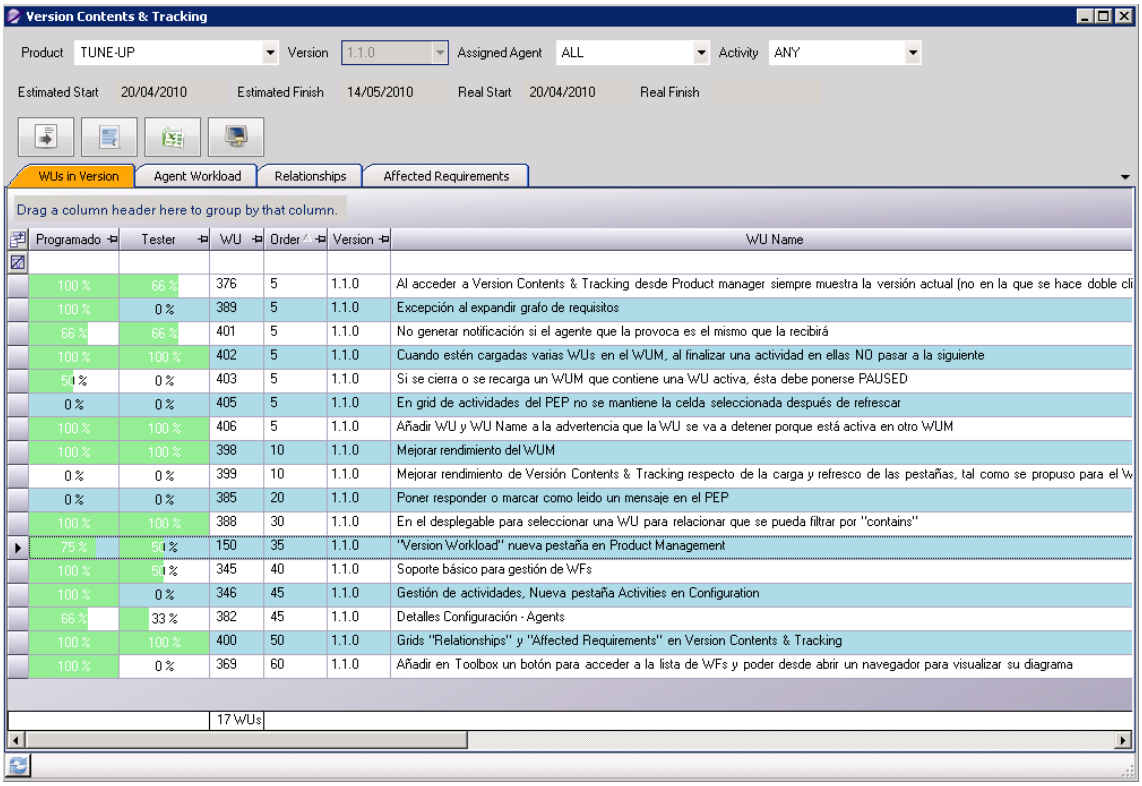


Figura 75. Planificador de Versiones – Pestaña Unidades de Trabajo en Versión

La ventana de la Figura 75 muestra la lista de unidades de trabajo asignadas a una versión de un determinado producto. El jefe de proyecto puede consultar los datos resumidos de cada unidad de trabajo en la versión, en particular, puede conocer la actividad actual en que se encuentra, el orden de prioridad, el esfuerzo que implica elaborar la unidad de trabajo, el agente asignado a las actividades principales del workflow, etc.

A su vez ofrece información a los agentes sobre el trabajo asignado y al jefe de proyecto la posibilidad de realizar un seguimiento sobre la carga de trabajo de los agentes, como podemos ver en la Figura 76.

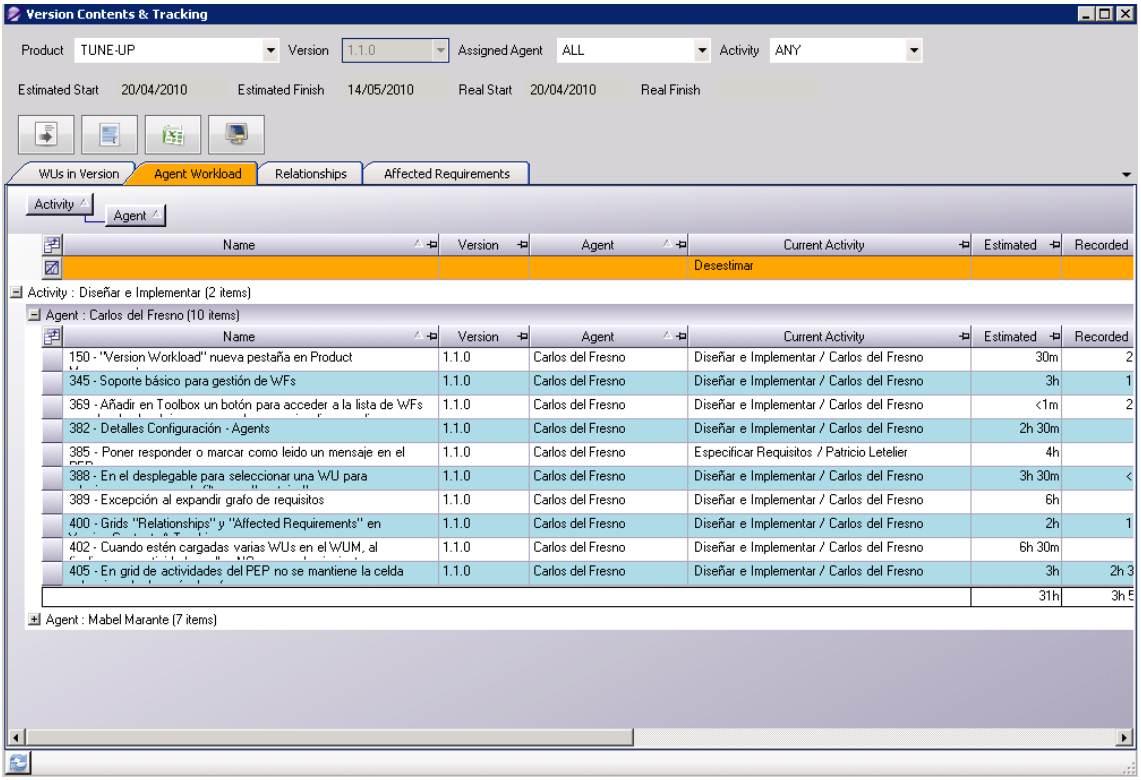


Figura 76. Planificador de Versiones – Pestaña Carga de Agentes

Todo esto ayuda al jefe de proyecto a tomar decisiones correctivas como reasignar y dividir unidades de trabajo o modificar plazos de entrega.

6.3 Mecanismos de colaboración

Una de las características más relevantes de TUNE-UP son los mecanismos de colaboración que existen: mensajes a uno o más destinatarios y reuniones. La colaboración es necesaria para disipar dudas, confirmar interpretaciones o tomar decisiones. Estos mecanismos de colaboración se aplicarán de acuerdo con las necesidades específicas en cada caso, y siempre considerando las siguientes pautas:

- **Mensaje**. Se trata de una consulta a uno o más agentes y que puede ser planteada (tanto en pregunta como en respuesta) de forma escrita. En este caso siempre cabe evaluar la conveniencia de un Mensaje en vez de una Reunión. La ventaja de las Mensajes es que no generan interrupciones para los receptores, pues se supone que se toman un tiempo conveniente para ir atendiéndolas evitando en lo posible postergaciones. La prioridad a la hora de contestar cada

petición debe ser analizada por el agente que tiene que ser capaz de establecer qué cosas son más prioritarias en cada momento.

- **Reunión**. Si se tiene que invertir demasiado tiempo en explicar por escrito, probablemente lo mejor es discutirlo cara a cara en una reunión, aunque esto necesitará igualmente una explicación introductoria (la convocatoria de la reunión con la posible documentación) y posteriormente una explicación detallada a modo de Memorando o resultado de la reunión. Según esto, una reunión permite agilizar la explicación y consenso respecto de algún aspecto cuando se estime que una interacción escrita no sea lo más efectivo. Por otra parte, cualquier colaboración tipo de consulta deberá registrarse como una reunión aunque sólo intervengan dos agentes.

6.3.2 Utilización de los saltos

La metodología considera la posibilidad de volver una unidad de trabajo hacia una actividad previa en el proceso para corregir posibles defectos que se hayan podido suceder en algún punto de la unidad de trabajo y así poder corregirlos. Esto implicará un re-trabajo en la actividad destino y posiblemente en las actividades posteriores que se habían finalizado. Una unidad de trabajo que "vuelve atrás" a una determinada actividad contabiliza un "reintento" en esa actividad. Cuando finaliza dicha actividad el agente tiene las mismas posibilidades de continuación que tiene cualquier actividad, es decir: seguir el Workflow o dar un salto (hacia adelante o hacia atrás). Cuando se trata de un reintento el agente debería de seguir el proceso cuando considere necesario rehacer el trabajo de las actividades que están entre la actividad que se ha reintentado y la actividad que originó el salto atrás. Si el cambio que ha introducido el reintento de una actividad no se considera relevante para las actividades posteriores (hasta la actividad que originó el salto atrás) entonces se debería hacer un salto adelante hacia la actividad que originó el salto atrás. Para realizar un salto atrás debería de tratarse de un "cambio importante" en la unidad de trabajo generada en la actividad a la cual se saltará. Entendemos por "cambio importante" aquel cambio que no permita continuar el trabajo de una actividad posterior o que pueda tener impacto en actividades ya finalizadas. Si no es un "cambio importante", se puede hacer una Petición al agente de la actividad (asociando dicha actividad a la petición) que presente el defecto para que éste lo solucione.

6.3.3 Estado de la Actividad

A través del seguimiento que se puede hacer de las unidades de trabajo es posible saber el estado actual de cualquier actividad, pudiendo encontrarse entre alguna de las siguientes:

- **Pendiente.** La actividad aún no ha sido comenzada por el agente.
- **Activa.** El agente está contabilizando tiempos en la actividad.
- **Pausada.** La actividad ha sido empezada por el agente pero en este instante no se están registrando tiempos.
- **Finalizada.** La actividad ha sido finalizada por el agente.

6.3.4 Cambio de Workflow

Una vez que la unidad de trabajo ha sido empezada es posible empezar cambiar el Workflow asociado, para ello la actividad en la que se encuentra la unidad de trabajo debe estar en el Workflow destino.

6.3.5 Cambio de Agente en un Actividad

En cualquier actividad del Workflow se puede realizar un cambio de agente, para ello hay que finalizar la actividad que se quiere cambiar de agente y seleccionar como actividad destino la actual y entonces nos aparecerá un nuevo desplegable donde seleccionar el agente que queramos que realice la actividad.

6.3.6 Trabajo en Paralelo

Una de las características que más flexibilidad le da a TUNE-UP es la posibilidad que dos agentes trabajen en paralelo con una misma actividad.

Capítulo 7. Workflows en TUNE-UP

Una vez presentada la metodología de TUNE-UP y su herramienta, vamos a pasar a describir el módulo específico de Workflow.

7.1 Descripción del sistema

El sistema que se va a describir pertenece a TUNE-UP Software Process, cuyo núcleo es un motor de Workflows el cual se encarga del correcto orden de las tareas, el flujo de las unidades de trabajo, la asignación de recursos, etc.

La interacción con los propios Workflows se divide en dos apartados: consulta y edición. En el caso de la consulta, el sistema actual se realiza mediante unos archivos generados mediante Microsoft Visio para la visualización de los Workflows desde el programa de gestión de tareas. La edición se realiza a través de la herramienta, mediante una tabla en la que cada registro es una actividad en el workflow asignándole una posición y un operador. Por ello, una vez definido todas las actividades del workflow, se irán desencadenando según su posición. Los Workflows están definidos en la base de datos por las tablas que aparecen en la Figura 77.

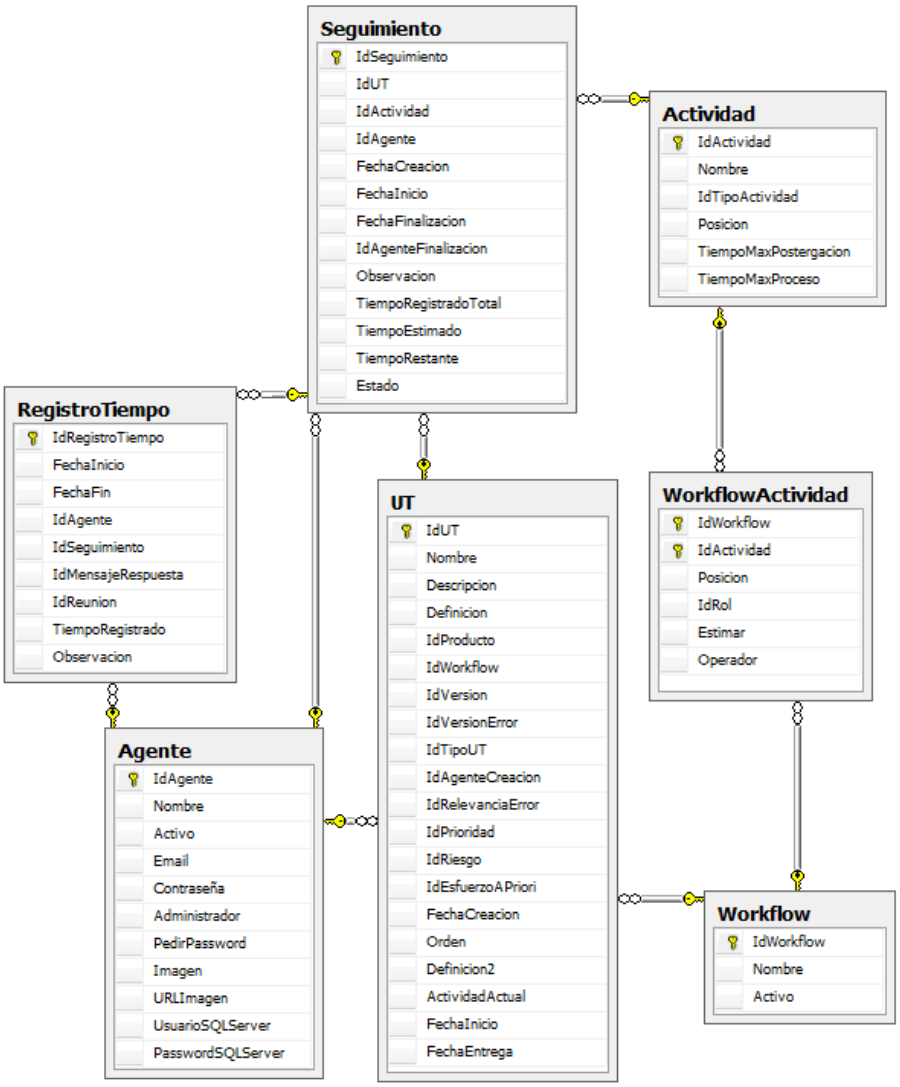


Figura 77. Diagrama de tablas involucradas en la gestión de workflows

7.2 Definición de workflows

Las actividades se asocian en workflows a través de la tabla WorkflowActividad. Para hacer que el workflow siga el flujo que nosotros queremos, hay que tener en cuenta los atributos de Operador y la Posición. El campo Operador es de tipo entero y puede tomar tres valores distintos:

- Secuencia: cuando dos actividades van seguidas una de la otra (Figura 78).

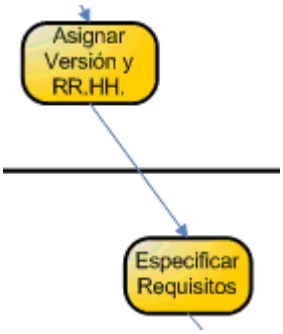


Figura 78. Secuencia de actividades

- Selección: cuando una unidad de trabajo puede tomar varios caminos o cuando dos caminos se unen (Figura 79).

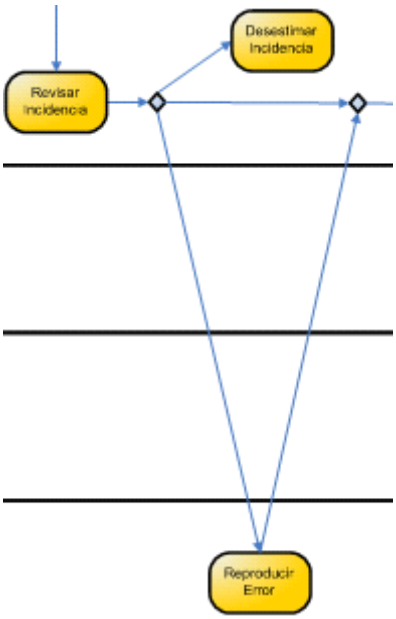


Figura 79. Selección de actividades

- Paralelo: cuando una unidad de trabajo se divide, realizándose a la vez, dos caminos distintos. A su vez también sirve para volver a unirlos (Figura 80).

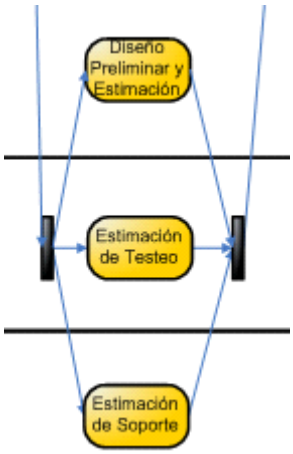


Figura 80. Paralelo de actividades

Los tres tipos de datos se representan utilizando, como hemos dicho anteriormente, los campos Operador y Posición. El convenio de representación de los Workflows queda reflejado en la base de datos de la siguiente manera:

- Secuencia: el Operador a elegir es "Secuenque" y la Posición debe ser una respecto al resto de actividades en el workflow. Como vemos en la definición del workflow de la Figura 81, se trata de un workflow en el que todas las actividades están en secuencia, por lo que la posición es diferente en cada uno de los estados y el Operador elegido es "Secuencia".

Position	Operator	Activity	Estimate	Role
1	Sequence	Introducir UT	☐	
2	Sequence	Asignar Versión y RRHH	☐	Product Manager
3	Sequence	Especificar Requisitos	☐	Analista
4	Sequence	Diseño Preliminar y Estimación	☐	Programador
5	Sequence	Confirmar Versión y RRHH	☐	Product Manager
6	Sequence	Diseñar e Implementar	☑	Programador
7	Sequence	Publicar	☐	Programador
8	Sequence	Aplicar Pruebas de Aceptación	☐	Tester
9	Sequence	Terminar	☐	

Figura 81. Definición de un workflow en secuencia

- Selección: conexiones de transiciones diferentes sin sincronización que tienen el mismo origen. Como vemos en la definición del workflow de la Figura 82, existe una selección de tres actividades, utilizando la misma "Posición" y el Operador "Selection", lo que quiere decir que, una vez terminada la actividad "Preparar Archivos" se dará a elegir al agente qué actividad desea realizar y una vez que esta se realice, se pasará a la siguiente de después de la selección, es decir, "Control de Calidad".

Position	Operator	Activity	Estimate	Role
1	Sequence	Introducir UT	☐	
2	Sequence	Recibir y confirmar	☐	Product Manager
3	Sequence	Preparar Archivos	☐	Product Manager
4	Selection	Traducir Francés	☐	Traductor Francés
4	Selection	Tradución Inglés	☐	Traductor Inglés
4	Selection	Traducción Alemán	☐	Traductor Alemán
5	Sequence	Control de Calidad	☐	Product Manager
6	Sequence	Preparación archivos cliente	☑	Product Manager
7	Sequence	Terminar	☐	

Figura 82. Definición de un workflow con una selección

- Paralelo: conexiones con mismo origen y de una misma transición sin sincronización. Como vemos en la definición del workflow de la Figura 83, existe un paralelo de tres actividades, las cuales comienzan después de finalizar la actividad de "Preparar Archivos" y, una vez terminadas las tres actividades en paralelo, se pasa a la actividad de "Control de Calidad". Como vemos, en el paralelo las actividades tienen la misma posición y además tiene el operador "Parallel".

Position	Operator	Activity	Estimate	Role
1	Sequence	Introducir UT	☐	
2	Sequence	Recibir y confirmar	☐	Product Manager
3	Sequence	Preparar Archivos	☐	Product Manager
4	Parallel	Traducir Francés	☐	Traductor Francés
4	Parallel	Tradución Inglés	☐	Traductor Inglés
4	Parallel	Traducción Alemán	☐	Traductor Alemán
5	Sequence	Control de Calidad	☐	Product Manager
6	Sequence	Preparación archivos cliente	☑	Product Manager
7	Sequence	Terminar	☐	

Figura 83. Definición de un workflow con un paralelo

7.3 Asignación de Workflows a productos

Una vez seleccionado un producto, se pueden asociar/desasociar workflows (Figura 84) según las necesidades, teniendo en cuenta que no se puede desasociar uno si existen unidades de trabajo no finalizadas que tengan dicho workflow.

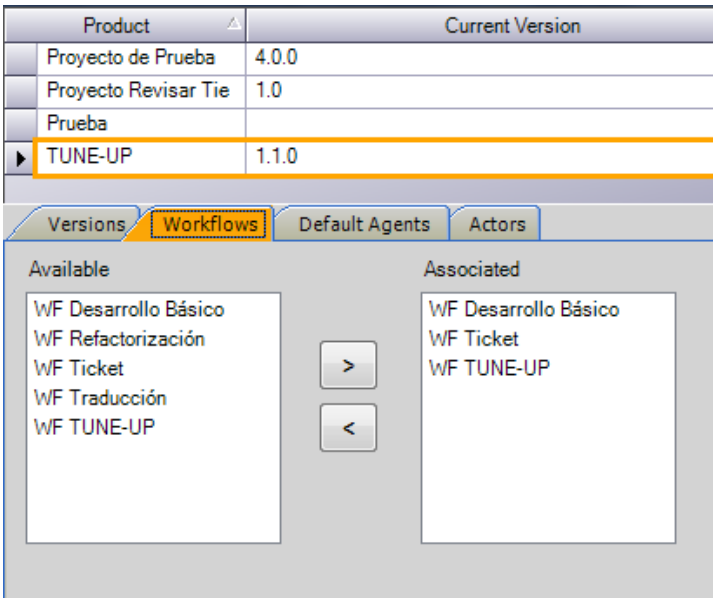


Figura 84. Asociación/desasociación de workflows a productos

7.4 Asignación de Agentes por defecto

Una vez asociado un workflow a un producto, hay que asignar los agentes por defecto de ese producto para ese workflow para que al crear nuevas unidades de trabajo se cojan esos agentes.

En la Figura 85 vemos un workflow que aún no tiene agentes asignados, por lo que en este caso, al crear unidades de trabajo del producto TUNE-UP con el WF Desarrollo Básico, se utilizará para hacer las actividades el agente que introdujo dicha unidad de trabajo.

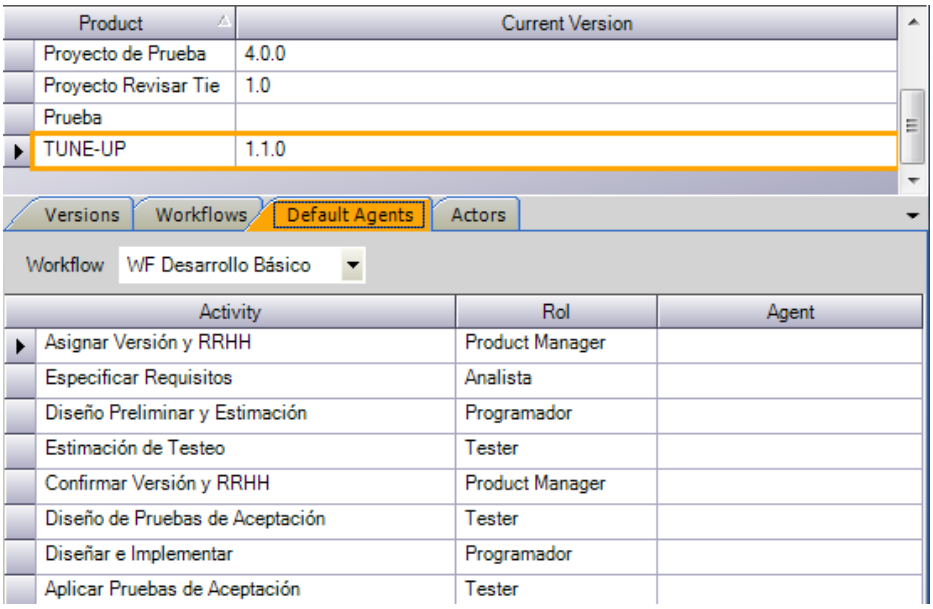


Figura 85. Asignación de agentes a productos en un workflow

En la Figura 86 vemos que para el producto TUNE-UP y WF TUNE-UP sí que están asignados, por lo que se utilizarán los agentes seleccionados para realizar las actividades.

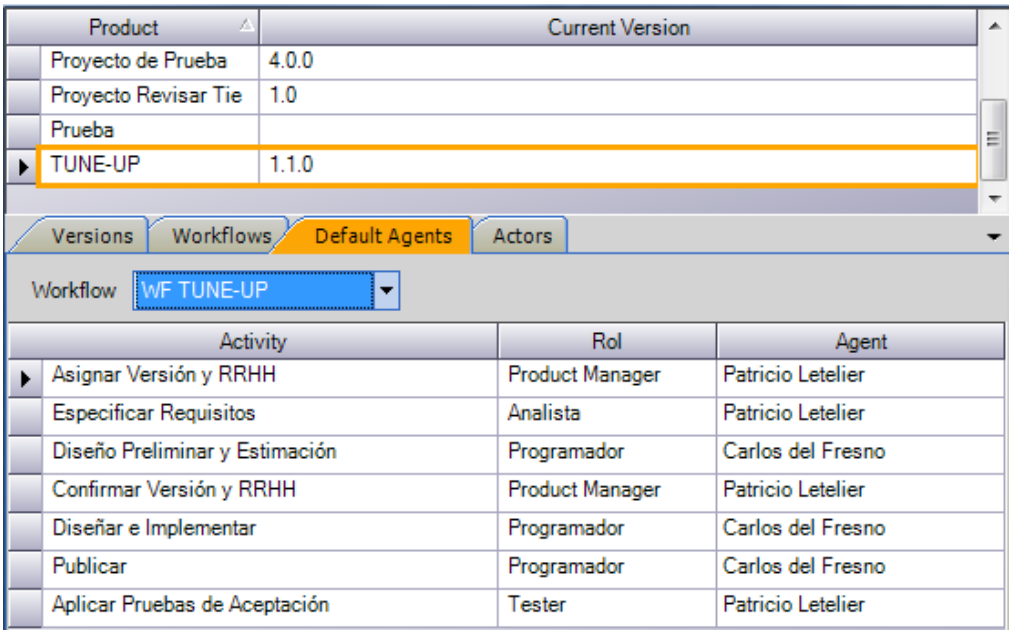


Figura 86. Asignación de agentes a productos en un workflow

7.5 Asignación de Agentes por unidad de trabajo

Como hemos dicho anteriormente, al crear una unidad de trabajo de un producto en un workflow determinado, se utilizan los agentes por defecto. En la Figura 87 podemos ver una unidad de trabajo del producto TUNE-UP y el WF TUNE-UP y al entrar en la

pestaña de Agentes Asignados, vemos que están todos los agentes, pudiendo cambiar según las necesidades.

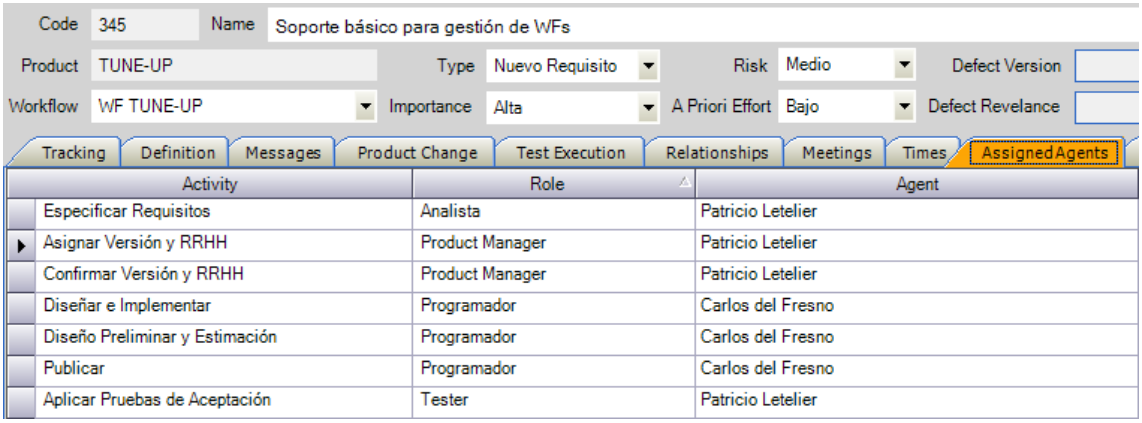


Figura 87. Asignación de agentes a una unidad de trabajo

7.6 Visualización en el PEP de la actividad-estado de una unidad de trabajo

En el Planificador Personal, tenemos una visión general del trabajo del agente conectado. En este caso, en la Figura 88 remarcado con un recuadro rojo, podemos ver todas las actividades pertenecientes a los workflows en los cuales el agente tiene trabajo pendiente junto con la contabilización de cuántas unidades de trabajo hay tanto en estado pendiente como en progreso. A la derecha, podemos ver una lista de seguimientos asociados a las actividades y las unidades de trabajo.

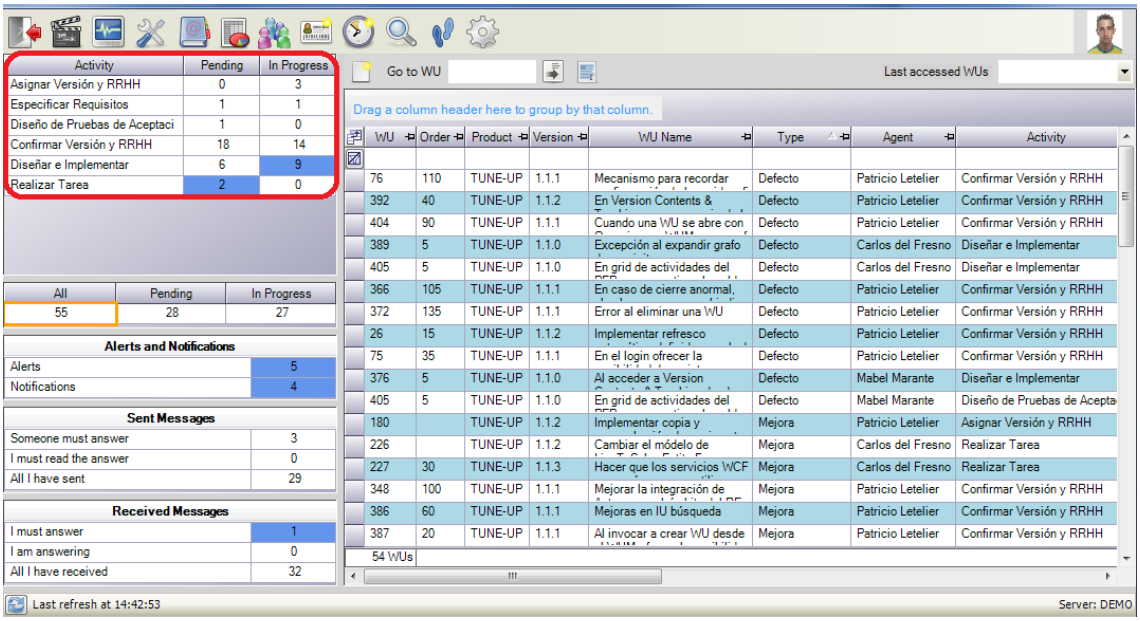


Figura 88. Visualización en el PEP del estado general de un agente

7.7 Visualización en VCT de la actividad –estado de una unidad de trabajo

A nivel de versiones en un producto, podemos ver todas las unidades de trabajo que están incluidas. En la Figura 89 podemos ver los detalles de la versión 1.1.0 del producto TUNE-UP viendo el avance de cada una de las unidades de trabajo, cuál es la actividad actual y a qué agente está asignada además de los tiempos registrados hasta el momento.

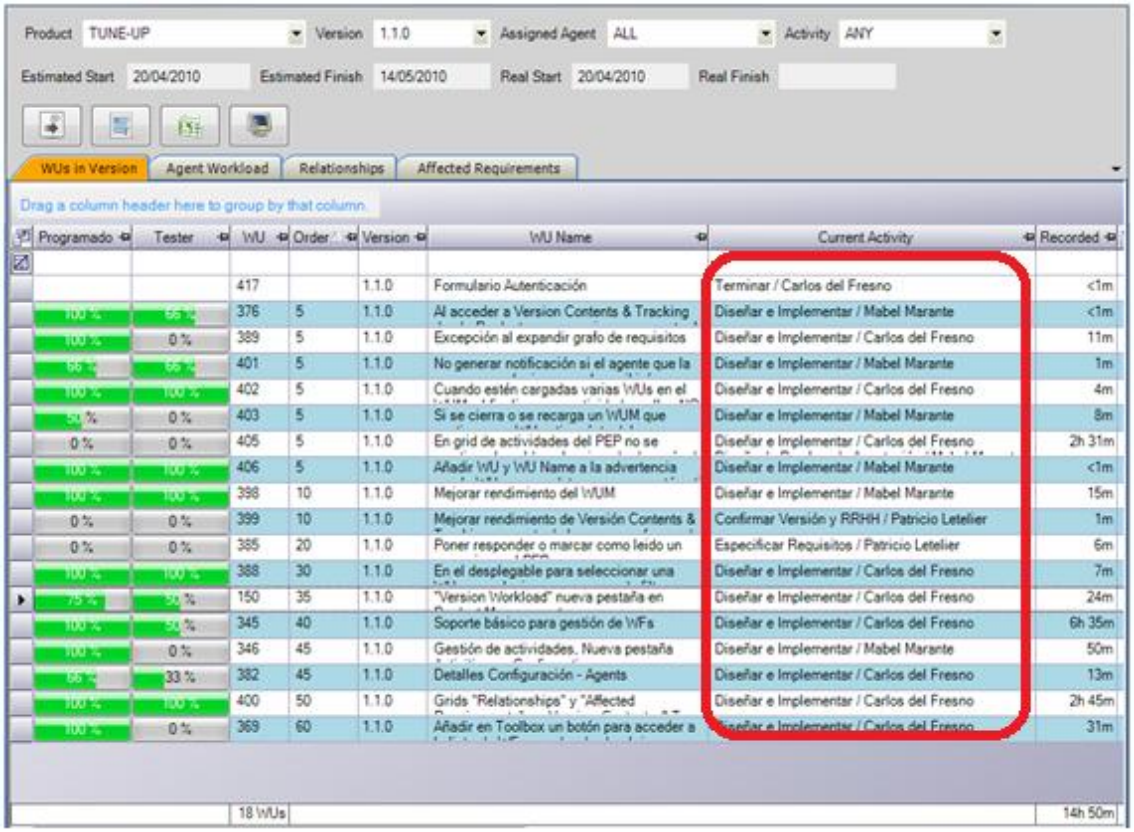


Figura 89. Visualización en el VCT del estado general de una versión

7.8 Registro de tiempos

Una de las características del motor de workflows es que se ha integrado con otras funcionalidades como el balanceo de la carga de agentes que se consigue a través de la interfaz de la Figura 90 en la que podemos ver, en base a una versión de un producto, la carga que tiene cada uno de los participantes, qué carga tiene y cuál es la carga en cada momento del proyecto. En la Figura vemos que tenemos los tiempos estimados (lo que estimó el agente que tardaría en realizar la actividad), el registrado (lo que lleva actualmente) y el restante (lo que le queda para llegar al estimado). A través de esta información, se puede hacer un balanceo para igualar la carga de trabajo entre varios

participantes y así asegurarse de que se cumplen los plazos de entrega y de que algún participante no quede ocioso.

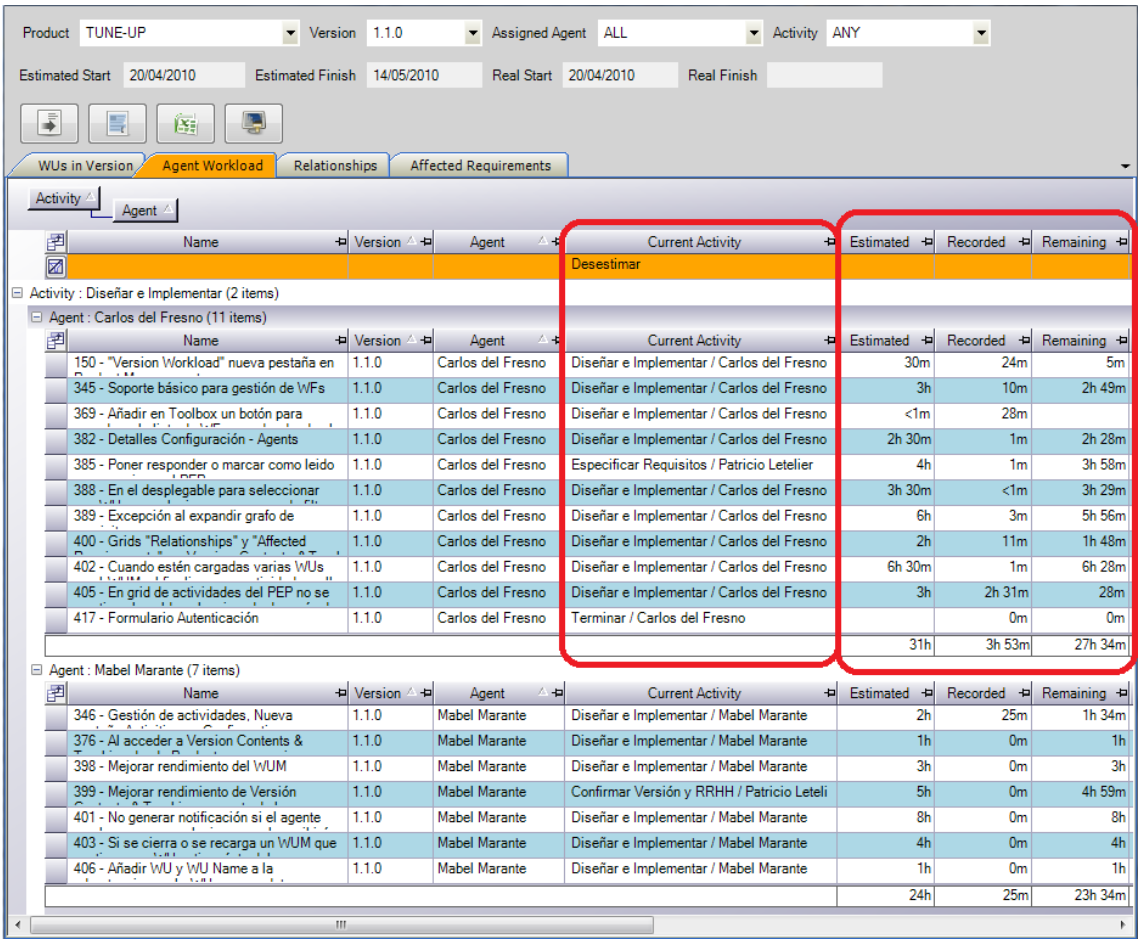


Figura 90. Visualización de una versión a nivel de agentes

7.9 Dashboard

A través de los registros de tiempo y otras características de lo workflows, se ha conseguido realizar un dashboard (Figura 91) con la información obtenida día a día, realizando una "foto" del estado actual del sistema, añadiendo Eventos de qué es lo que ha ocurrido en comparación con el día anterior, como por ejemplo: nuevas unidades de trabajo añadidas/quitadas a/de una versión, asignación de una actividad de una unidad de trabajo a otro agente, re-estimación de una actividad, etc.

Con estos datos es posible averiguar, por ejemplo, a través de gráficas, cómo va el proyecto y saber si se van a cumplir los plazos y en el caso de que no se cumplan, averiguar la causa, como podría ser que se hayan añadido unidades de trabajo a última hora en la versión.

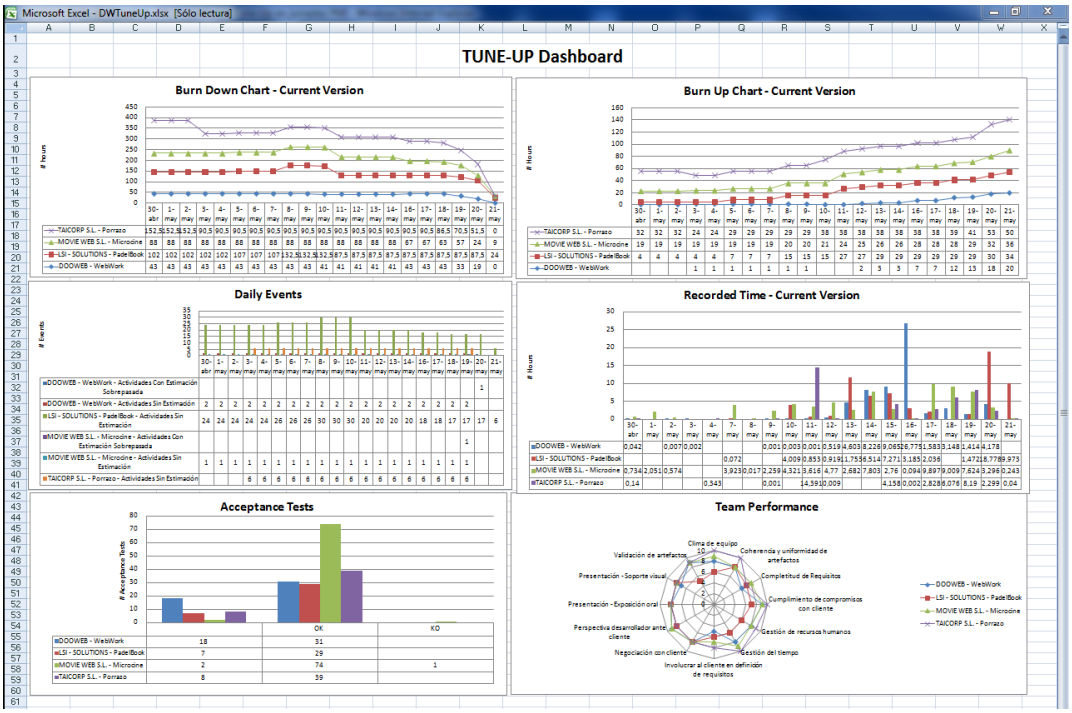


Figura 91. Dashboard

7.10 Situaciones excepcionales tratadas por código

A continuación se van a tratar todas las situaciones que no se han podido tratar por el sistema actual de Workflow por no cubrir ese tipo de situaciones y que han tenido que ser tratadas por código. Todas las situaciones que se van a reflejar son del WF General, por lo que en el anexo se añadirá una captura del Workflow generado mediante Microsoft Visio.

7.10.1 Situación 1

Si los checks de "Incluir en ACT" y "Actualizar Ayuda" no están activados, la actividad "Actualizar Ayuda" no se crea bajo ningún concepto.

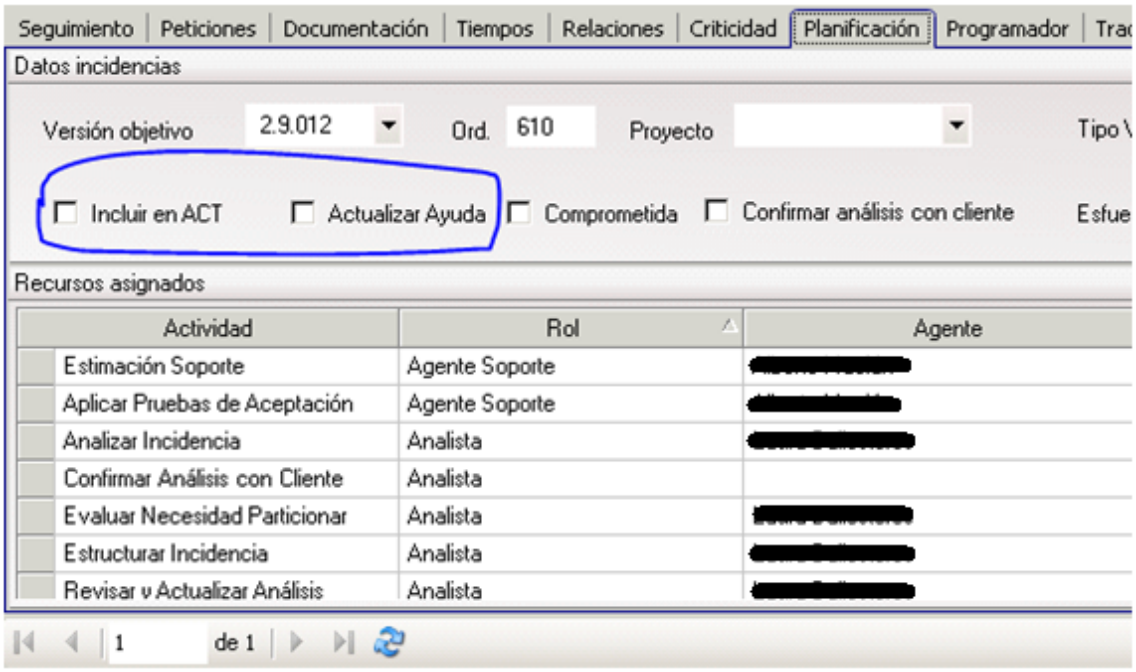


Figura 92. Situación 1

7.10.2 Situación 2

Si no está marcado ninguno de los checks de la pestaña Traducción (Figura 93), al ir de la actividad "Diseño e Implementación" a la siguiente, las actividades de traducción no se abren, sino que va directamente a "Aplicar Pruebas de Sistema" (en el caso de que no haya ninguna actividad en paralelo, en caso contrario se esperará a que estén finalizadas).

Figura 93. Situación 2

7.10.3 Situación 3

Si se realiza un salto de "Diseño e Implementación" a "Aplicar Pruebas de Sistema" (Figura 94) y hay actividades en paralelo, no bloqueará como hacía antes sino que se finalizará y se quedará esperando a que las que hay en paralelo se finalicen. Si no hay ninguna en paralelo se realiza el salto. Esto antes no podía suceder debido a una

restricción en la que no se puede realizar un salto hacia adelante si existen actividades en paralelo abiertas, por lo que por esta modificación es posible realizar el salto.

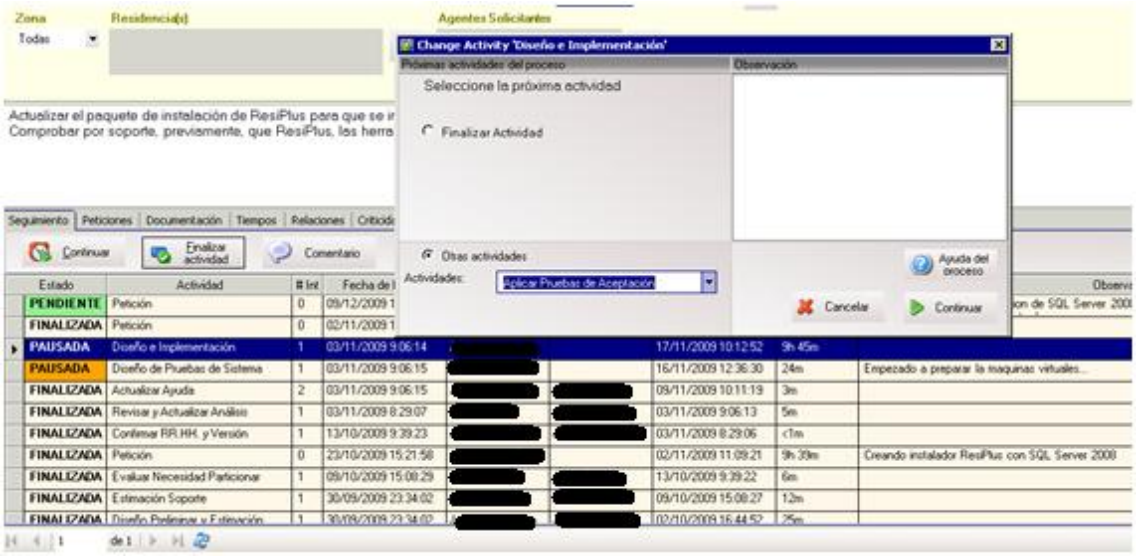


Figura 94. Situación 3

7.10.4 Situación 4

Si finalizamos la actividad "Validar Análisis" y en la pestaña de Planificación no está marcado en check "Confirmar análisis con cliente" (Figura 95), se saltará la actividad "Confirmar Análisis con Cliente" e irá directamente a la posterior.

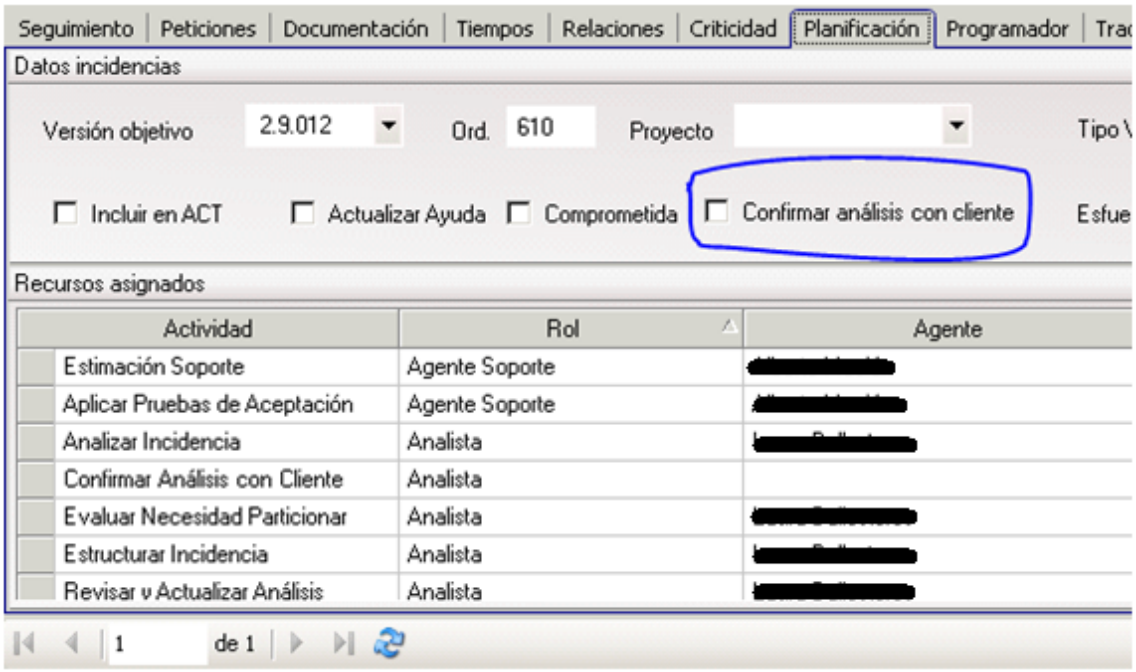


Figura 95. Situación 4

7.11 Frameworks Utilizados

Para la generación del motor de Workflows se han utilizado dos frameworks, que van a comentarse brevemente a continuación.

7.11.1 CodeSmith

CodeSmith [6] es una herramienta de generación de código fuente basado en plantillas que puede utilizarse para cualquier lenguaje de programación que use texto ASCII.
Esta herramienta presenta su propio IDE enfocado al desarrollo de plantillas de código, y dispone también de opciones para la generación de código desde la línea de comandos.
Las plantillas de CodeSmith tienen una sintaxis muy similar al código de ASP.NET, pudiendo utilizar la sintaxis de cualquier lenguaje (C#, Java, VB, PHP, ASP.NET, SQL, etc.).

CodeSmith incluye muchas plantillas útiles, así como conjuntos completos de plantillas para la generación de arquitecturas probadas (.NetTiers, CSLA, NHibernate, PLINQO, Wilson's ORMapper, APOSA y más). También es posible modificar cualquiera de las plantillas o escribir una nueva para poder generar el código de una forma en concreto.

7.11.2 .NetTiers

.netTiers es un conjunto de plantillas de código libre que simplifica las tareas de crear las capas de aplicación para las aplicaciones en .NET en tan sólo unos minutos [22].

.netTiers utiliza el potencial de una potente herramienta de generación de código llamada CodeSmith [6]. La arquitectura generada por .netTiers está personalizada al dominio al que se quiere aplicar, usando patrones de diseño y sigue la guía

recomendada de Microsoft de patrones y prácticas [33]. De hecho, la arquitectura base de .netTiers está construida sobre Microsoft Enterprise Library Application Blocks.

7.11.2.1 Características

- Genera una solución de Visual Studio completamente compilable, junto con proyectos distintos con cada una de las capas para la aplicación.
- Crea un conjunto completo de procedimientos almacenados especializados para el dominio de la aplicación. Este código puede ser ejecutado en código como parámetros de SQL y así no ser utilizado como procedimientos almacenados.
- Genera automáticamente objetos de entidad y su relación como objetos de un dominio basado en las tablas de la base de datos.
- Las clases incluyen tanto las clases parciales como las clases concretas, por lo que se puede personalizar la lógica de las capas generadas.
- Utiliza una lista genérica personalizada para las colecciones que soporta todas las interfaces ComponentModel.
- Crea un proyecto de sitio web completo, ya pre-configurado y listo para comenzar la programación.
- Crea un conjunto completo de controles web de administración, que sirve de base de administración web completamente funcional de la consola de base de datos.

7.11.3 Manual de generación de código mediante .NetTiers y CodeSmith

A continuación se van a presentar los pasos a seguir para hacer la generación de código de la base de datos.

1. Abrir la plantilla NetTiers.cst que se incluye al instalar CodeSmith (Figura 96).

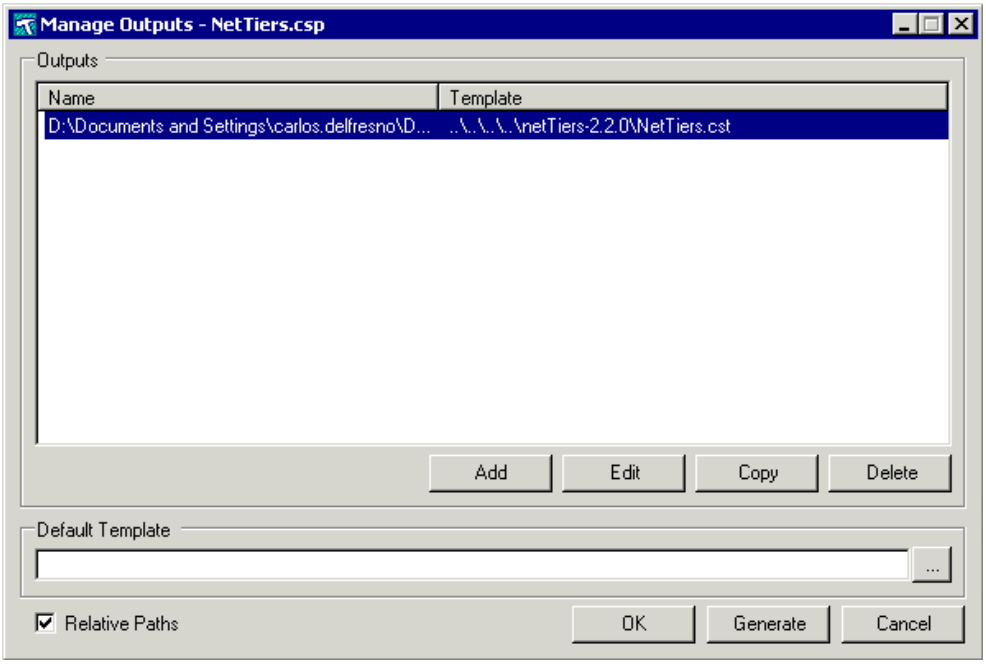


Figura 96. Elegir la plantilla

2. Presionamos en Edit para personalizar la generación (Figura 97).

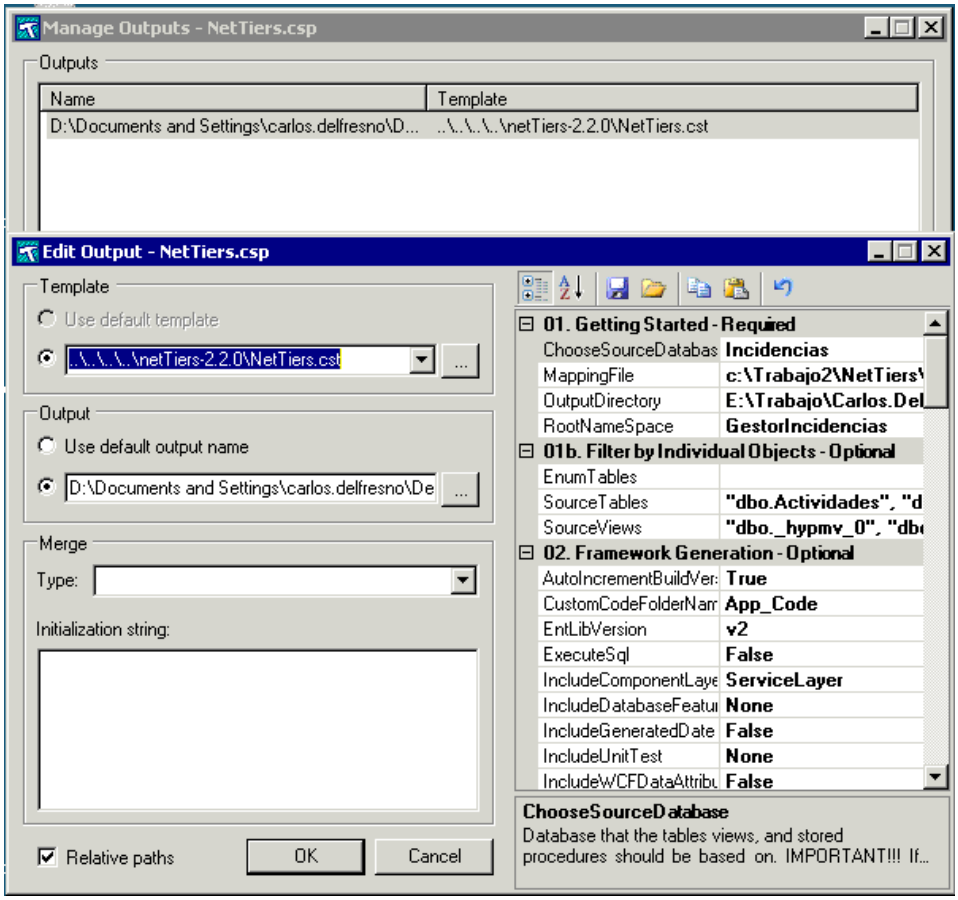


Figura 97. Opciones de la generación de código

3. En esta ventana podemos configurar todos los parámetros. A continuación detallamos los más importantes.

- ChooseSourceDatabase. El origen de datos a parsear.
- OutputDirectory. El directorio de salida de los resultados.
- RootNameSpace. El espacio de nombres raíz para las clases C# generadas.
- EnumTables. Las tablas a generar como enumeraciones.
- SourceTables. Las tablas que se desean generar.
- SourceViews. Las vistas que se desean generar.
- BusinessLogicLayerNameSpace. El espacio de nombres que se añade al de raíz de la capa de entidades.
- ComponentLayerNameSpace. El espacio de nombres que se añade al de la raíz de la capa de negocio.
- DataAccessLayerNameSpace. El espacio de nombres que se añade al de la raíz de la capa de acceso de datos.
- UnitTestsNameSpace. El espacio de nombres que se añade al de la raíz para las clases de pruebas.

4. Una vez configurado, le damos a Ok y luego a Generate (Figura 98 y 99).

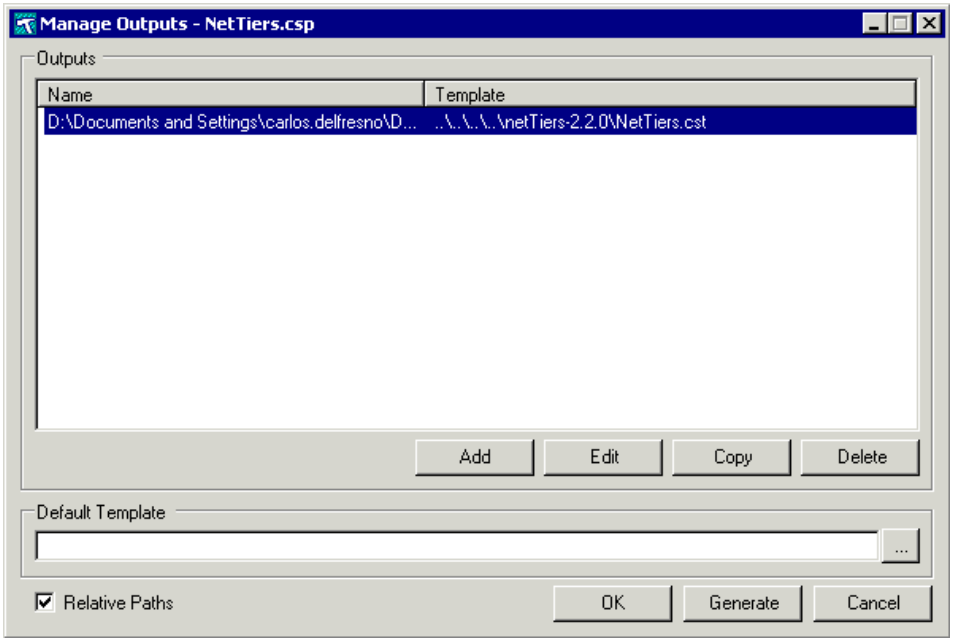


Figura 98. Elegir la plantilla

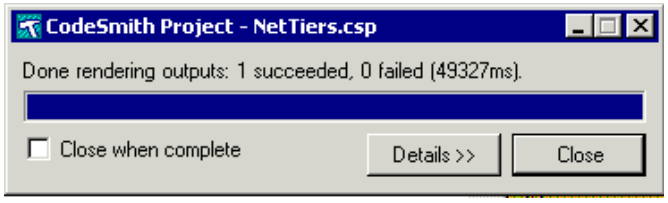


Figura 99. Barra de progreso de la generación

5. Una vez terminado el proceso de generación, se abre una página web (Figura 100) con el informe del parseado en el que podremos ver si ha habido algún error.

Figura 100. Informe de éxito de la generación de código

6. Ahora ya se puede empezar a trabajar con la solución generada.

7.11.3.1 Re-generación de la base de datos por modificaciones posteriores

En el caso de que hayamos generado en un primer momento la base de datos y posteriormente, después de crear la aplicación queramos añadir ciertos campos o tablas en la base de datos habrá que hacer de nuevo el parseado. Para ello se seguirán los siguientes pasos.

1. Se seguirán todos los pasos citados en el apartado anterior.

2. Abrir la nueva solución generada y hacer una búsqueda por proyecto en el orden en que aparece en la solución para que sea más ordenado buscando la tabla nueva o el nuevo campo a añadir.
3. Por cada fichero encontrado, añadirlo a nuestra solución.
4. Si el fichero ya existe, reemplazarlo.
5. Nunca realizar el reemplazo de ficheros que las clases parciales, ya que en estas clases es donde se añade lógica adicional. Hay que realizar el reemplazo tanto de las interfaces como de las clases *generated*.

Capítulo 8. Workflows en Acción

8.1 Saltos

Para realizar un salto adelante hay que seleccionar la actividad que queremos finalizar y presionar el botón de finalizar (Figura 101).

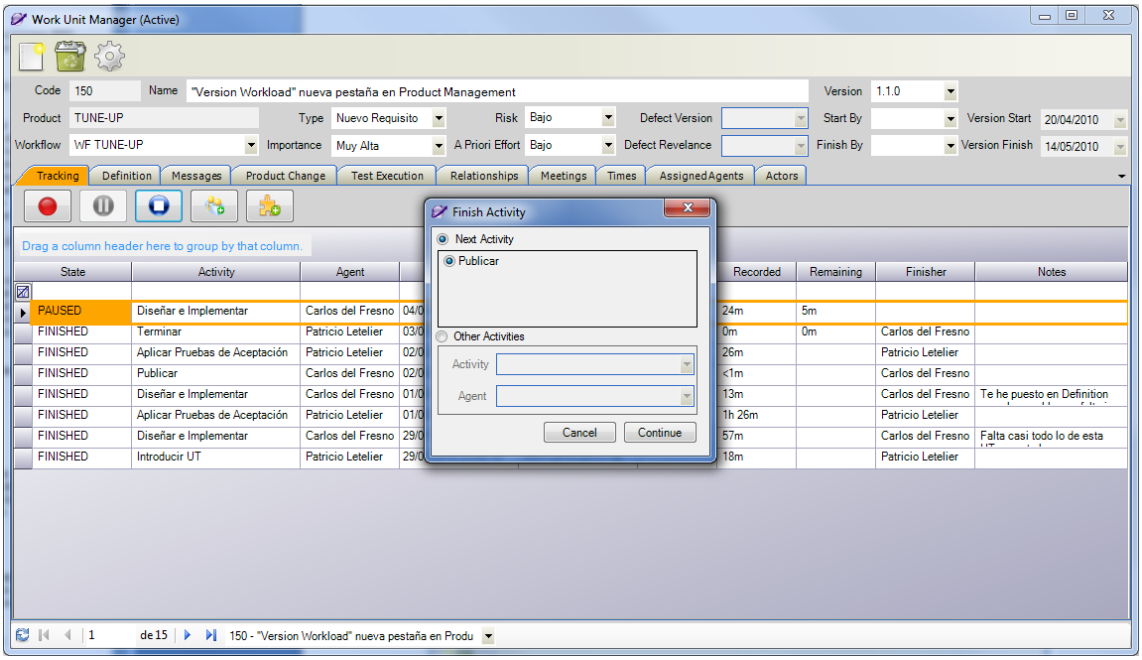


Figura 101. Finalización de actividad

Una vez que nos salga el formulario de Cambio de Actividad tenemos que seleccionar "Otras Actividades" en el panel inferior izquierdo y de esta manera se nos activará el desplegable "Actividad" (Figura 102).

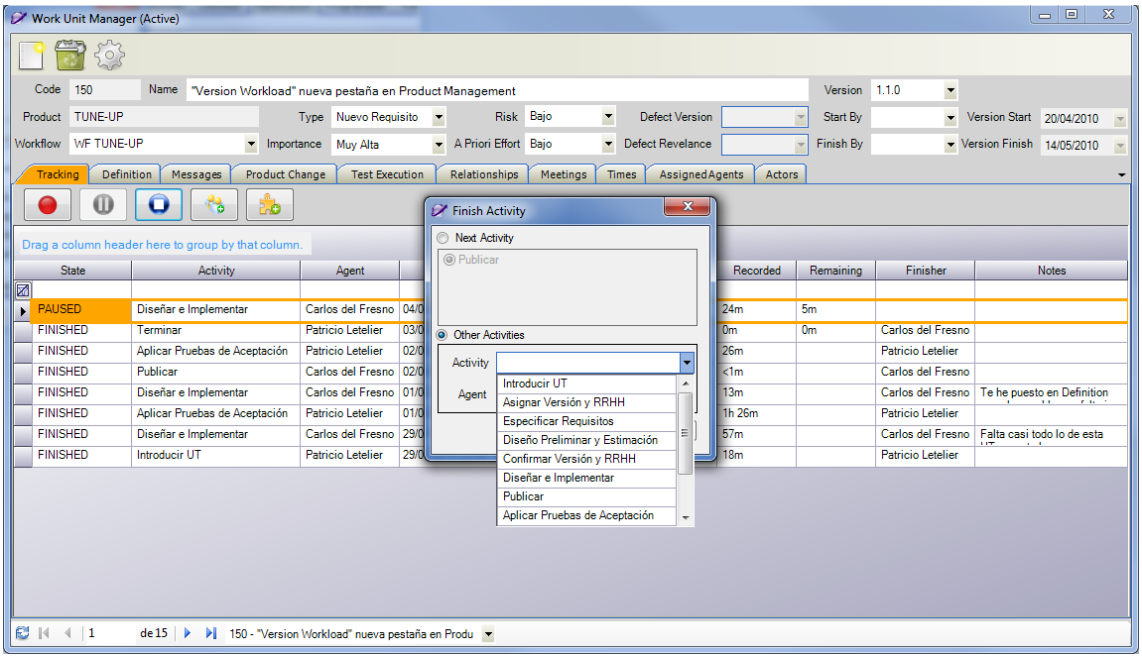


Figura 102. Selección de actividad de salto

En el desplegable podemos ver el orden de las actividades dentro del Workflow de la unidad de trabajo. Ahora, para realizar el salto hacia adelante o hacia atrás hay que seleccionar la actividad a la cuál queremos saltar. Como ejemplo, seleccionaremos "Confirmar Versión y RRHH" y presionaremos el botón "Continuar" para realizar el salto, en este caso hacia atrás.

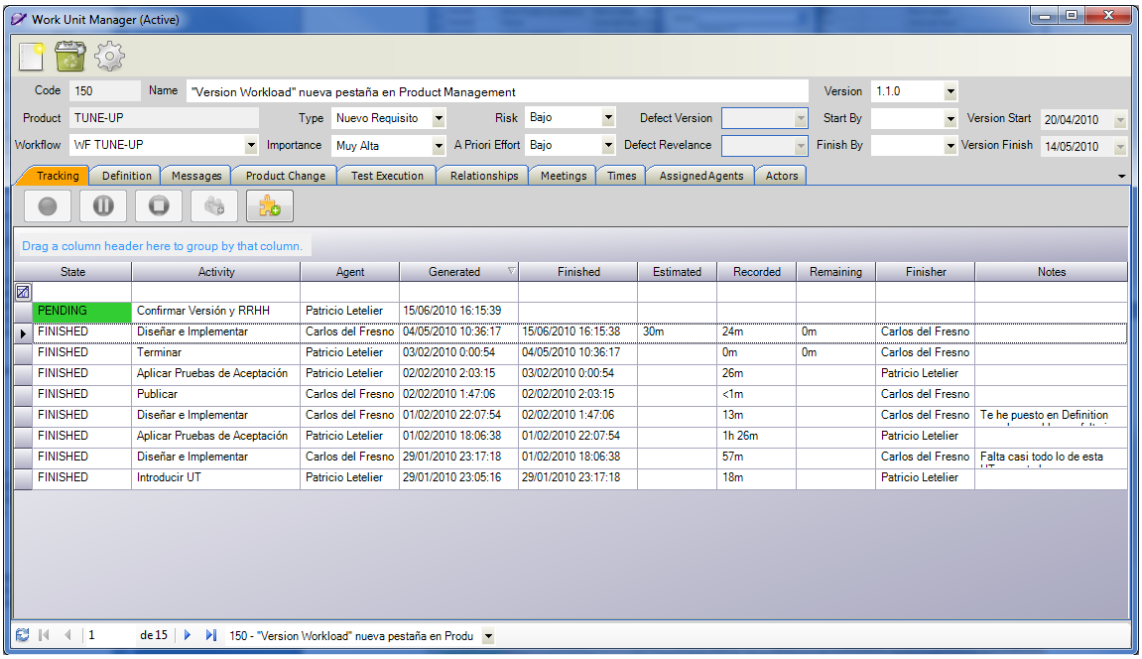


Figura 103. Seguimiento tras el salto

Y podemos ver (Figura 103) cómo se ha abierto un nuevo seguimiento con la actividad destino en estado Pendiente y cerrando previamente la actividad anterior.

8.2 Cambio de Agente

Para realizar un cambio de agente, hay que ir a la pestaña de "Agentes Asignados" y para la actividad en la que queremos hacer el cambio, seleccionamos el nuevo agente y, en el caso que tengamos la actividad seleccionada abierta, la cerrará y la abrirá con el nuevo agente. En el ejemplo de la Figura 104, la actividad de "Diseñar e Implementar" la tiene el Carlos del Fresno, por lo que vamos a realizar un cambio de agente a Mabel Marante.

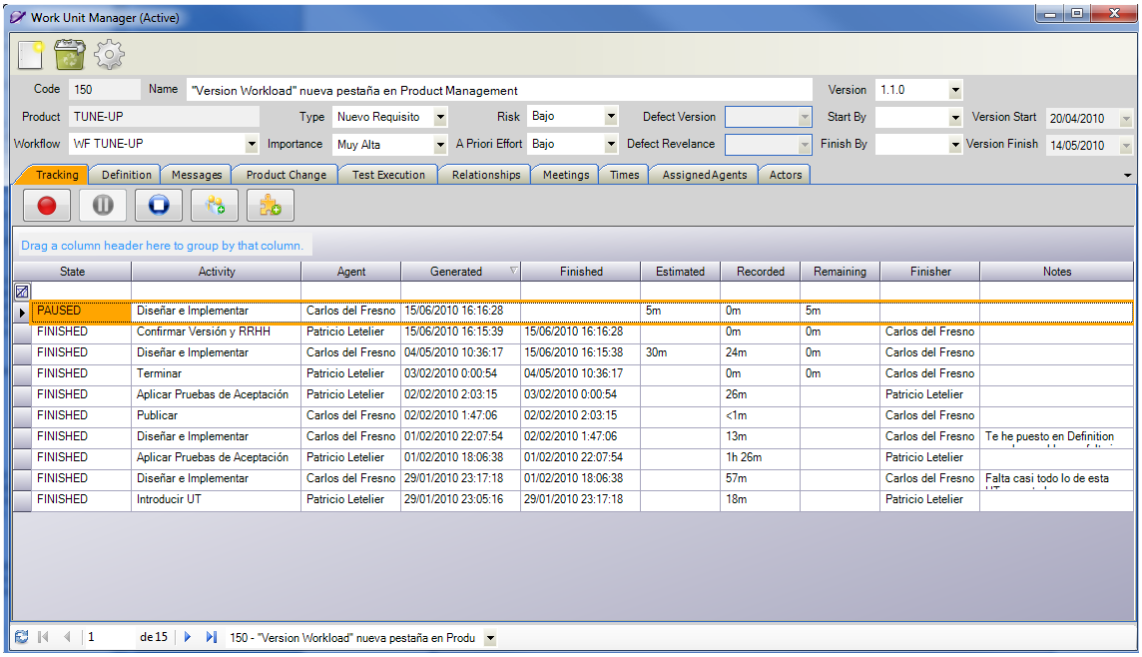


Figura 104. WUM con actividad en pause

Por ello, en la Figura 105 se muestra la pestaña de Agentes Asignados en la cual vamos a cambiar el agente.

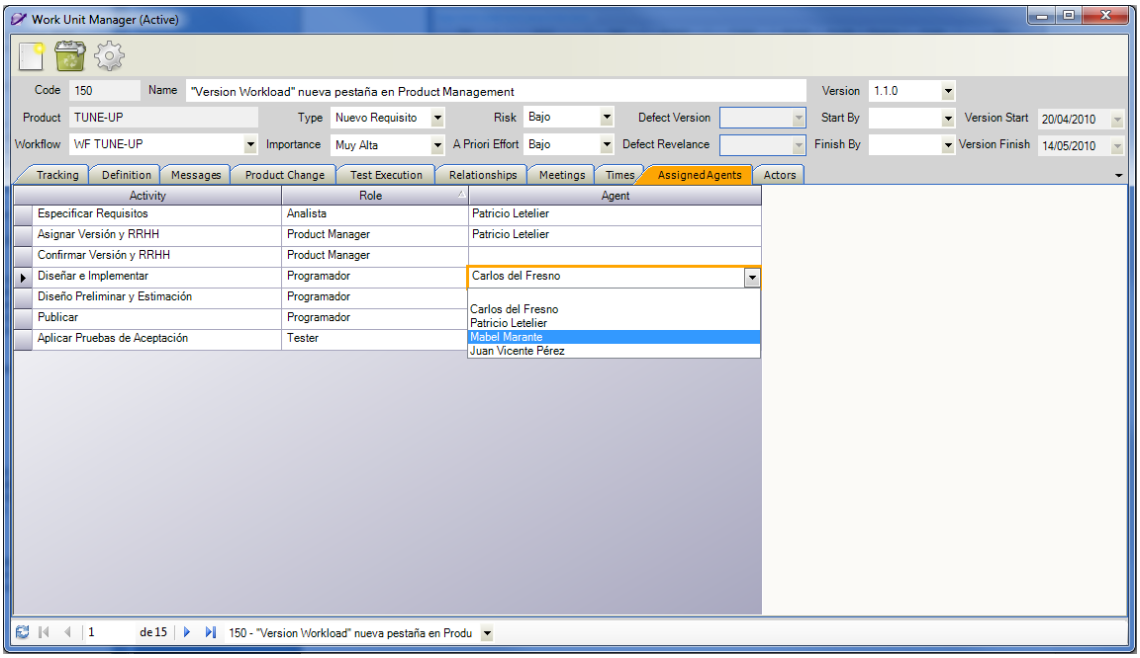


Figura 105. Cambio de agente asignado

Una vez aplicado el cambio, volvemos a la pestaña de Tracking y vemos en la Figura 106 que el cambio se ha aplicado correctamente, abriéndose un nuevo seguimiento con el nuevo agente asignado.

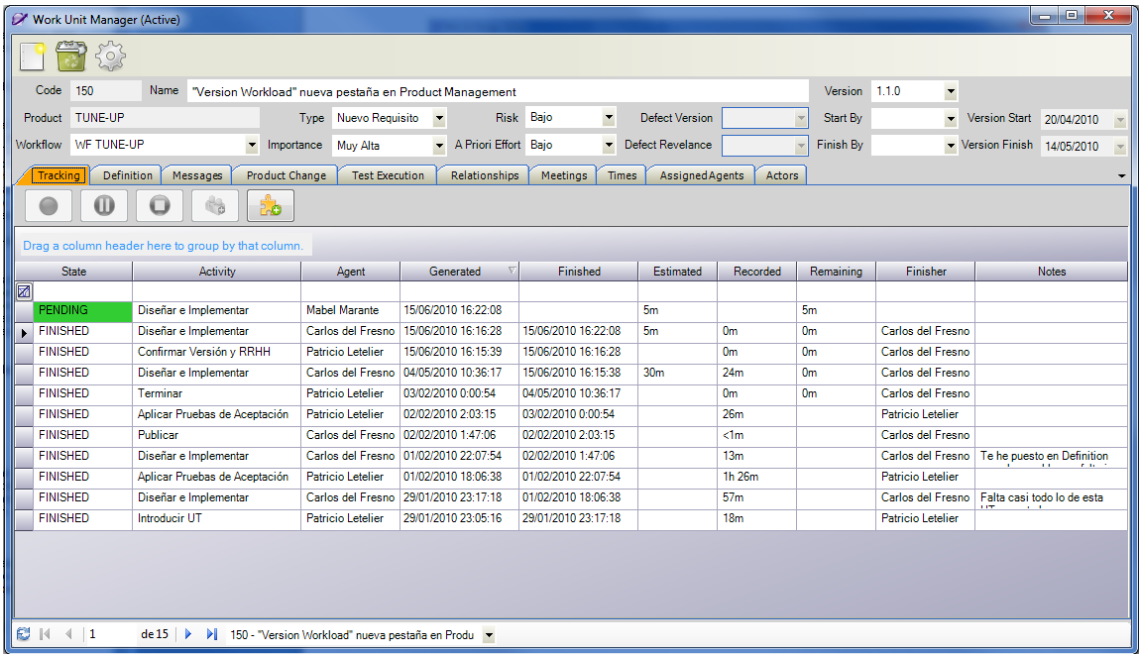


Figura 106. Unidad de trabajo una vez hecho el cambio de agente

8.3 Trabajo en paralelo

Se pueden añadir actividades adicionales pero cambiando de agente para así poder trabajar en paralelo. Para ello, hay que seleccionar la actividad con la cual se quiere

trabajar en paralelo y presionar el botón marcado en rojo en la Figura 107 y nos saldrá un formulario para seleccionar el agente que se quiere para trabajar en paralelo, con algunas validaciones como que el agente no tenga ya una actividad abierta con la misma.

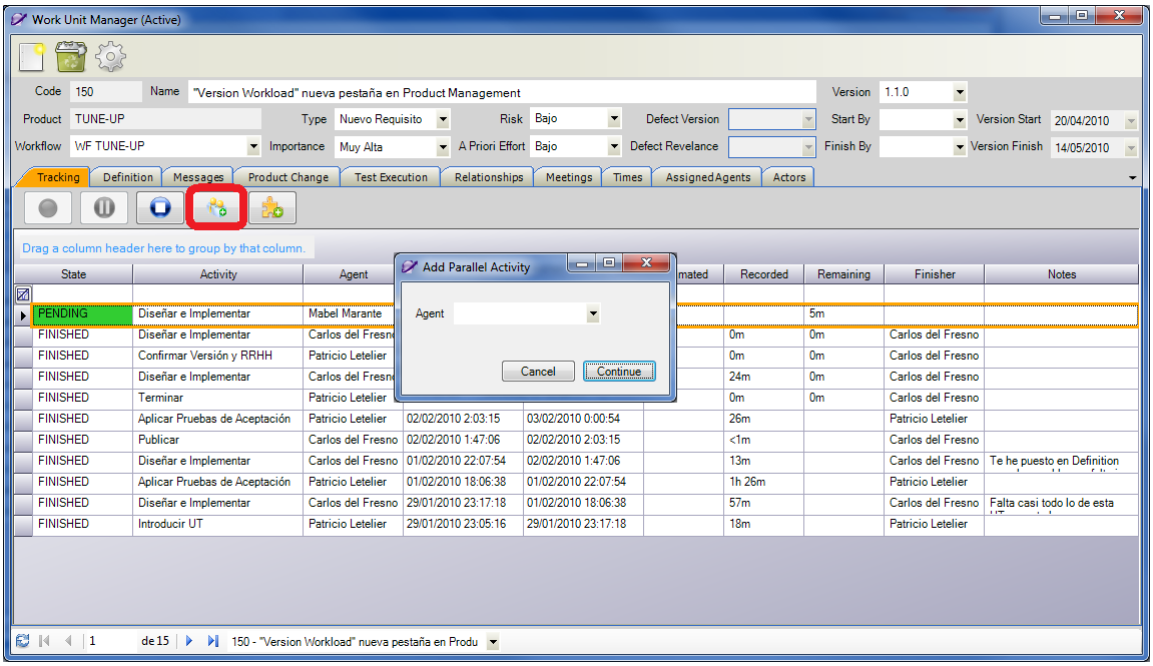


Figura 107. Añadir actividad en paralelo

Al seleccionar un agente, se abrirá un nuevo seguimiento con la misma actividad como podemos ver en la Figura 108, teniendo ambas en paralelo pero con diferentes agentes y hasta que las dos no estén finalizadas, no se seguirá el flujo normal del workflow.

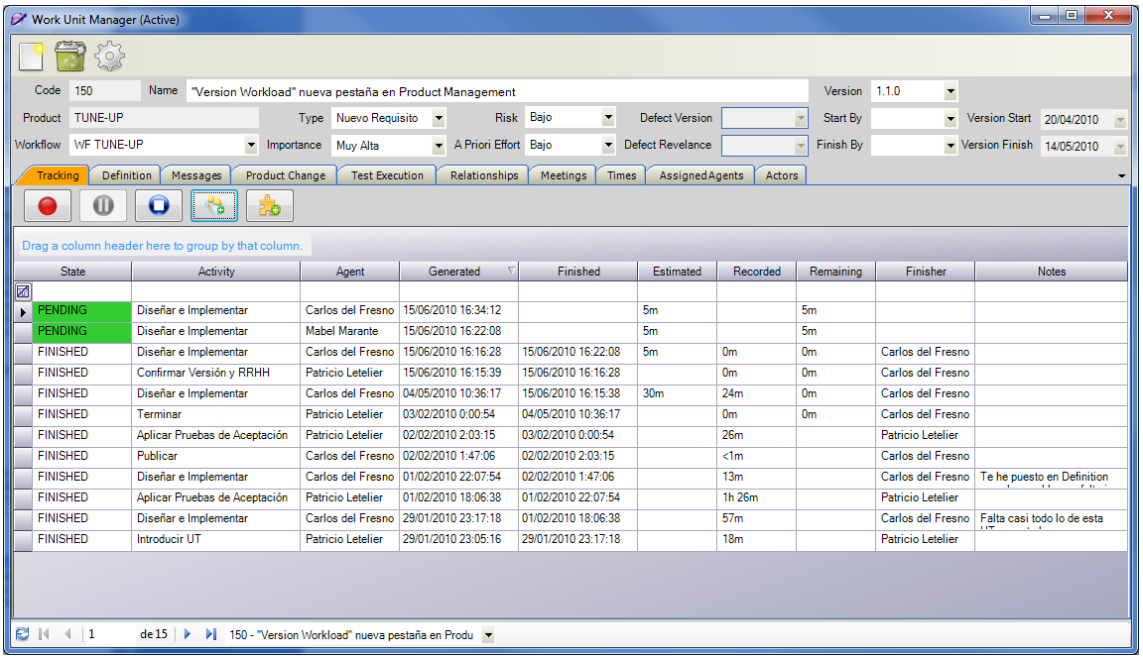


Figura 108. Unidad de trabajo con dos actividades en paralelo

8.4 Añadir nuevas actividades

También es posible añadir actividades que inicialmente no están en el workflow de la unidad de trabajo. El motivo de querer introducir una actividad nueva pueden ser diversos, como por ejemplo, que sea algo muy puntual y que no valga la pena crear un workflow para ese cambio.

Para añadir, hay que presionar el botón marcado en rojo en la Figura 109 y se nos mostrará un formulario para seleccionar una actividad y un agente.

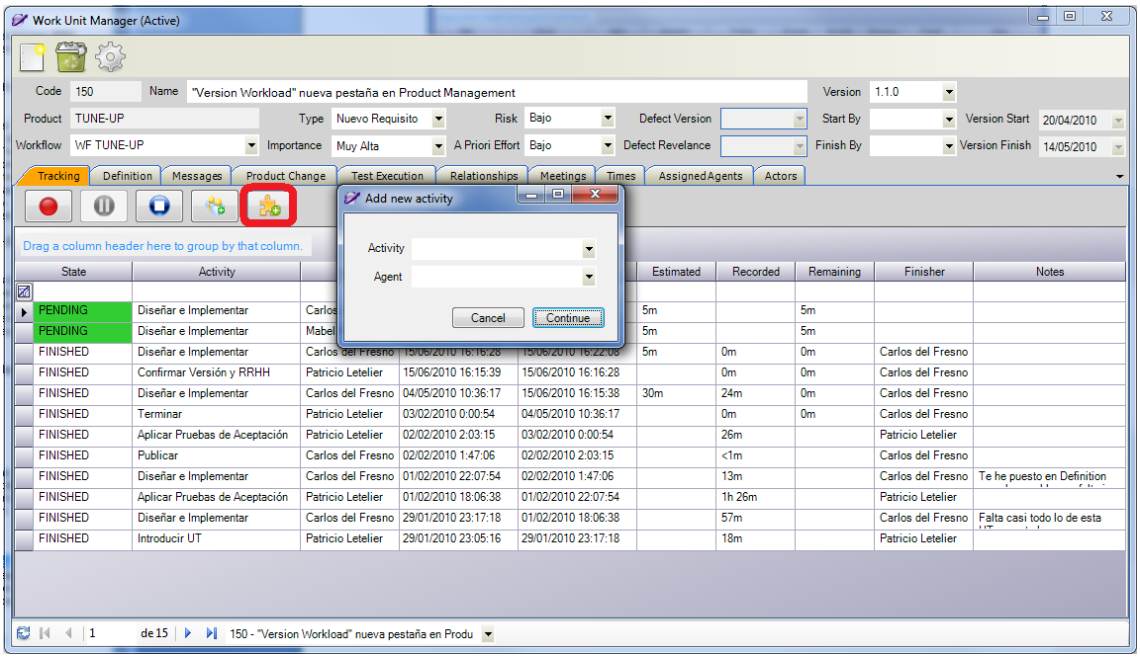


Figura 109. Añadir actividades que no están en el workflow de la unidad de trabajo

Una vez seleccionado la actividad y el agente, se creará un nuevo seguimiento como vemos en la Figura 110 el cual podrá finalizarse en el momento en el que el agente finalice su tarea, y además, no se tiene en cuenta esta nueva actividad para seguir el workflow. La única restricción es que no se puede finalizar la unidad de trabajo si existen actividades sin finalizar.

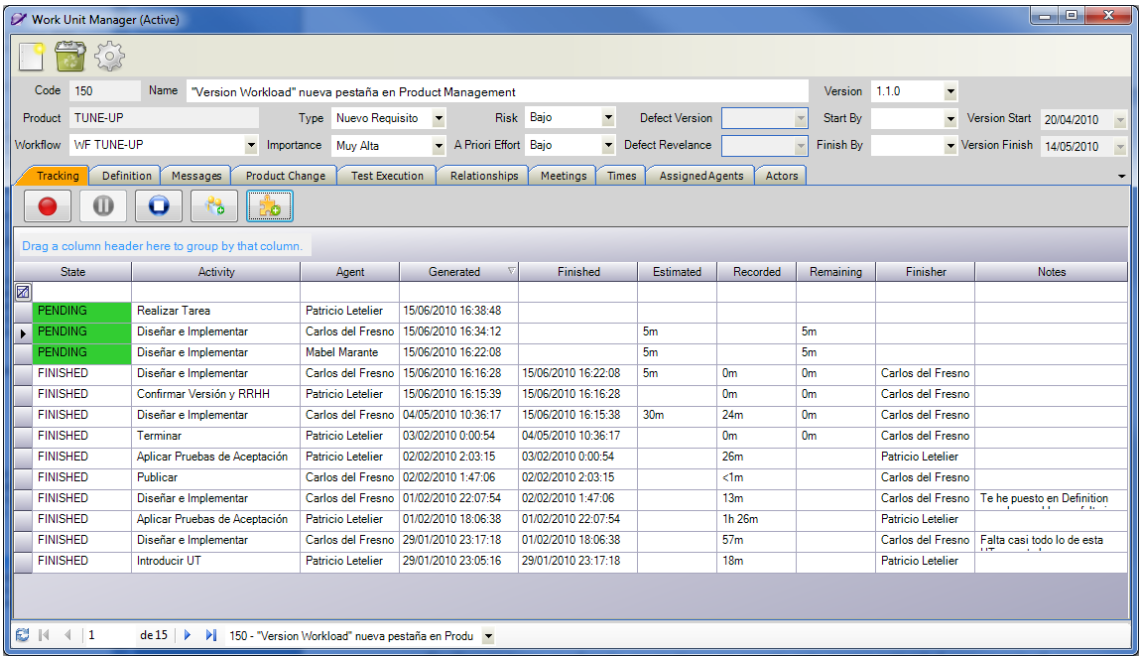


Figura 110. Unidad de trabajo con actividades no pertenecientes al workflow

Capítulo 9. Conclusiones y trabajo futuro

Para conseguir altos niveles de productividad, eficiencia y calidad en gestión de proyectos es necesaria la utilización de metodologías y herramientas que ayuden a la planificación y al seguimiento de los mismos. Considerando el proceso de desarrollo de software como un proceso, una manera efectiva de gestionar este proceso es mediante la utilización de workflows. Los workflows en desarrollo software deben ser flexibles y actuar como guías de trabajo ofreciendo información sobre el estado del desarrollo y los siguientes pasos a realizar, todo desde una perspectiva de representación explícita del conocimiento y madurez de los procesos del equipo de desarrollo.

Los requisitos de flexibilidad para los workflows en TUNE-UP Software Process llevaron a desarrollar un motor de workflows sencillo pero a la vez potente en cuanto a posibilitar muchas acciones no frecuentes en workflows tradicionales, tales como:

- Saltos de actividad
- Trabajo en paralelo
- Añadir actividad que no están en el workflow de la unidad de trabajo
- Cambios de agente en actividad
- Cambios de workflow de una unidad de trabajo

Por otra parte, el motor de workflows se ha integrado en otras funcionalidades de TUNE-UP Software Process tales como:

- Asignación y balanceo de la carga de agentes
- Planificación y seguimiento de proyectos
- Mecanismos de comunicación
- Dashboard

Adicionalmente se ha desarrollado la gestión de workflows mediante la cual los workflows pueden ser editados en tiempo de ejecución.

Este motor ha sido comparado a posteriori con estándares para medir el grado de expresividad con otros motores existentes comparándolos contra una lista de patrones workflow definidos en [49].

En la tabla 4 se presenta una comparativa de los modelos presentados en cuanto a la representación de workflows, tal y como hemos visto en el capítulo 3. En esta tabla se

puede ver cómo los lenguajes soportan casi todos los patrones de Van der Aalst [50]. Únicamente patrones complejos como los de Hito (WPC-18), o Rutina Paralela Entrelazada (WPC-17) no pueden ser descritos por estos lenguajes. De los modelos orientados a la representación es el lenguaje XPDL el que mejor resultado ofrece, ya no por su capacidad expresiva, sino por su formato de fichero estándar lo que facilita la creación de motores de interpretación asociados. En cuanto a nuestro motor desarrollado, aunque implementa alrededor de siete patrones, no nos ha hecho falta que incluya más patrones ya que con estos han sido suficientes para conseguir la flexibilidad requerida.

Patrón	Motor TUNE-UP	BPMN	XPDL
WPC-1 (Secuencia)	+	+	+
WPC-2 (Separación Paralela)	+	+	+
WPC-3 (Sincronización)	+	+	+
WPC-4 (Elección Exclusiva)	+	+	+
WPC-5 (Fusión Simple)	+	+	+
WPC-6 (Multielección)	-	+	+
WPC-7 (Fusión Sincronizada Estructurada)	+	+	+
WPC-8 (Multifusión)	-	+	+
WPC-9 (Discriminador Estructurado)	-	+/-	+/-
WPC-10 (Ciclos Arbitrarios)	-	+	+
WPC-16 (Elección Aplazada)	+	+	+
WPC-17 (Rutina Paralela Entrelazada)	-	-	-
WPC-18 (Hito)	-	-	-

Tabla 4. Evaluación de los patrones de control de flujo básicos en los modelos de representación estudiados

Y en la tabla 5 se presenta la comparativa de los modelos basados en la interpretación de WF presentados en el capítulo 3 en cuanto a la expresividad.

Los modelos orientados a la interpretación por su naturaleza formal usualmente suelen tener recortadas sus posibilidades expresivas. A pesar de usar metáforas gráficas para mejorar la legibilidad de los modelos, estas son en muchas ocasiones demasiado próximas a los lenguajes imperativos, lo que en algunos casos se hace más dificultosa la legibilidad. Algunos modelos como el WWF intentan mejorar estos problemas proponiendo varias formas de diseñar los modelos. Entre ellas, usando modelos basados

en estados que son mucho más intuitivos para los expertos. Sin embargo, esta vista de diseño en el WWF es poco expresiva, ya que no permite la ejecución de estados paralelos.

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

Tabla 5. Evaluación de los patrones de control de flujo básicos en los modelos de interpretación estudiados

Como resultado y al igual que en la comparación con los lenguajes de representación de la tabla anterior, vemos el motor de workflows desarrollado cubre pocos patrones, pero que cubren las necesidades que en la práctica nos hemos enfrentado hasta el momento.

Por otra parte, comparándolo con la notación gráfica de BPMN, ofrece los siguientes operadores vistos anteriormente:

- Compuerta Exclusiva Basada en Datos
- Compuerta Exclusiva Basada en Eventos
- Compuerta Paralela
- Flujo de secuencia
- Tarea
- Eventos planos de comienzo y fin.

A pesar de que la cantidad de elementos comunes con BPMN que implementa el motor de workflows desarrollado no es muy elevada, hay que resaltar que se ha conseguido alcanzar la expresividad deseada.

Dicho motor lleva implementado y en funcionamiento cerca de tres años en una PYME de desarrollo de software bajo el marco de varios proyectos universidad-empresa y durante este tiempo dicho motor se ha ido refinando para adaptarse a las necesidades que se requería ofreciendo más flexibilidad.

En la implantación actual de TUNE-UP se tienen 20 workflows definidos, con algunos de ellos de hasta 30 actividades, siendo éstos un factor clave en el funcionamiento del departamento de desarrollo de la empresa.

Dentro de las futuras líneas de trabajo se tienen:

- Incremento de la expresividad, ya que ahora mismo solo nos podría estar interesando implementar es el anidamiento de operadores.
- Inyección de código para condiciones, para elegir automáticamente un camino del workflow, pre-condiciones para poder iniciar una actividad y post-condiciones para poder finalizar.
- Integración de un entorno gráfico para la edición de workflows ya que actualmente se realiza a través de un formulario de la herramienta.
- Explotar la información de los workflows con DataMining, para poder extraer información, por ejemplo, se puede explotar la información de los seguimientos que se generan de una unidad de trabajo para descubrir nuevos workflows.

Finalmente quisiera destacar la experiencia profesional que ha significado para mí el desarrollo de esta tesis. Durante dos años he estado trabajando con un equipo de trabajo desarrollando y mejorando la metodología TUNE-UP. Nos hemos enfrentado a un

marco de trabajo en el que cada tres semanas realizábamos entregas de funcionalidad de la herramienta implantada en la empresa. Además, hemos podido interactuar con los usuarios de la herramienta, y ofrecerles el soporte, entrenamiento y apoyo necesario. Actualmente, tanto la herramienta como la metodología que incluye el motor de workflows están fuertemente implantadas y resultan indispensables para organizar el trabajo de los agentes.

La realización de esta tesis, además de permitir al autor trabajar en un proyecto real en colaboración con una PYME de desarrollo de software, ha tenido que profundizar en tecnologías de workflows, en diferentes metodologías y las herramientas que las soportan. Además, haber participado en el desarrollo de TUNE-UP, una metodología y su herramienta de apoyo para el proceso de desarrollo de software que es la que integra el motor de workflows desarrollado en esta Tesis con el fin de una futura spin-off.

Referencias

1. Bizflow 2000. Handysoft. http://www.handysoft.com/products/products.asp
2. Blog de MSDN. http://msdnfan.blogspot.com/2006/07/conceptos-bsicos-de-windows-workflow.html
3. Blog de developer.com. http://www.developer.com/net/net/article.php/3627266/Building-a-Practical-Application-with-Windows-WorkFlow-Foundation
4. Booch, G., Rumbaugh, J., Jacobson, I., *El lenguaje de Modelado Unificado. Guía de usuario*, Ed. Addison-Wesley, 1999.
5. BPMN Poster. BPMN. http://bpt.hpi.uni-potsdam.de/pub/Public/BPMNCorner/BPMN1_1_Poster_EN.pdf
6. CodeSmith Tools. CodeSmith. http://codesmithtools.com
7. COSA Workflow. COSA Solutions. http://www.cosa.nl
8. *Creation and Deployment of a Process Definition*. RedHat. http://www.redhat.com/docs/en-US/JBoss_SOA_Platform/4.2.2/html/SOA_ESB_JBPM_Integration_Guide/ch01s04.html
9. David Chappell. Enterprise Service Bus. OReilly, 2004.
10. De Witt, D. et al., *El manifiesto del Sistema de Base de Datos Orientado al Objeto*, Novática Vol. XVII, núm 91, 1991.
11. DOLPHIN. Fujitsu. http://www.fnc.fujitsu.com
12. eiStream. Eastman Software Enterprise. http://www.eastmansoftware.com/products/
13. Enhydra Team. Enhydra shark site: http://www.enhydra.org/index.php.

14. Georgakopoulos, D., Hornick, M., Sheth, A., *An Overview of Workflow Management: from Process Modeling to Workflow Automation Infraestructure, Distributed and Parallel Databases*. Vol.3, n.2, April 1995.
15. Guía de instalación de jBPM. JBOSS. http://docs.jboss.org/jbpm/v3/spanish/jbpm-gpd-installation-spanish.pdf
16. Hollingsworth, D., *The Workflow Reference Model*, Technical report TC00-1003, WfMC, January, 1995. http://www.wfmc.org/
17. IBM MQSeries Workflow. http://www.redbooks.ibm.com
18. Lotus Notes. http://www.lotus.com/home.nsf/tabs/lotusnotes
19. Miguel Valdes et al. Bonita site: http://wiki.bonita.objectweb.org/xwiki/bin/view/main/.
20. Marante M, Letelier P, Suarez F. *TUNE-UP: Seguimiento de proyectos software dirigido por la gestión de tiempos*. XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2009). San Sebastián, Septiembre de 2009.
21. Marante M, Letelier P, Suarez F. TUNE-UP: *Un enfoque pragmático para la planificación y seguimiento de proyectos de desarrollo y mantenimiento de software*. I Congreso Iberoamericano SOCOTE -Soporte al Conocimiento con la Tecnología Universidad Politécnica de Valencia (SOCOTE 2009). Valencia, Noviembre de 2009.
22. .netTiers. http://nettiers.com/
23. N. Russell, A.H.M. ter Hofstede, D. Edmond, and WMP van der Aalst. *Workflow data patterns*. QUT Technical report, FIT-TR-2004-01, Queestand University of Technology, 2004.

24. N. Russell, A.H.M. ter Hofstede, D. Edmond, and WMP van der Aalst. *Workflow Workflow resource patterns*. Working Paper Series, WP 127, Edindhoven University of Technology, 2001.
25. N. Russell, A.H.M. ter Hofstede, WMP van der Aalst, and N. Mulvar. *Workflow control-flow patterns: A revised view*. BPM Center Report, 2006.
26. N. Russell, WMP van der Aalst, and A.H.M. ter Hofstede. *Exception handling patterns in process-aware information systems*. BPM Center Report, 2006.
27. Object Management Group. Workflow Management Facility: Request for Proposal.
28. ObjectManagement Group. Business Process Modeling Notation (BPMN). Specification. OMC dtc/06-02-01, 2006.
29. ObjectManagement Group. Business Process Modeling Notation (BPMN). Technical Report cf/97-05-06. July 1997. http://www.omg.org/library/
30. Object Management Group. Workflow Management Facility Specification. Revised Submission bon/98-06-07. July 1998. http://www.omg.org/library/
31. Object Management Group. Workflow Management Facility Specification, v1.2. Technical Report April, 2000. http://www.omg.org/library
32. Organization for the Advancement of Structured Information Standards. Oasis wsbpel standard: http://docs.oasis-open.org/wsbpel/2.0/os/wsbpel-v2.0-os.html
33. Patterns & Practices. Microsoft. http://msdn.microsoft.com/en-us/practices/default.aspx
34. Proyecto EXOTICA. http.//www.almaden.ibm.com/cs/exótica
35. Proyecto MENTOR: Middleware for Enterprise-wide Workflow Management. http://www.dbs.cs.uni-sb.de/~mentor/
36. Proyecto MOBILE. http://www6.informatik.uni-erlangen.de/research/wfm.html

37. Proyecto OPERA. http://www.inf.ethz.ch/deparment/IS/iks/research/opera.html

38. Proyecto WIDE. http://dis.sema.es/projects/WIDE/

39. Roger S. Pressman. *Ingeniería del Software: Un enfoque práctico*. McGraw Hill, 2 edition, 1999.

40. Russell, N., A.H.M. ter Hofstede, W.M.P. van der Aalst, and

41. SERFloware. SER. http://www.ser.de/en/products/products-main.html

42. Sheth, A. et al., *Report from the NSF Workfshop on workflow and Process Automation in Information Systems*. Computer Sicence Department Technical Report, UGA-CS-TR-96-003, University of Georgia, October 1996. http://lsdis.cs.uga.edu/activities/NSF-workflow/

43. Staffware Worfklow. Staffware Corporation. http://www.staffware.com

44. Stephen A. White, Introduction to BPMN. http://bpmn.org/Documents/Introduction_to_BPMN.pdf

45. TeamWare Flow. Fujitsu. http://www.fnc.fujitsu.com

46. TIB/InConcert. TIBCO. http://www.tibco.com/products/in_concert/

47. Visual and Panagon Workflow. FileNet Corporation USA. http://www.filenet.com/English/Products/index.asp

48. University of Technology Eindhoven and University of Technolog Queensland. Workflow patterns initiative: http://www.workflowpatterns.com/

49. WfMOpen Team. Wfmopen site: http://wfmopen.sourceforge.net/

50. Wil M. P. van der Aalst, Alistair P. Barros, Arthur H. M. ter Hofstede, and Bartek Kiepuszewski. *Workflow patterns. Distributed and Parallel Databases*, page 70, 2003. [105]

51. Windows Workflow Foundation. Microsoft. http://msdn.microsoft.com/en-us/netframework/aa663340.aspx (WWF1)

52. Workflow Management Coalition. Process Definition Interface – XML Process Definition Language. WFMC-TC-1025, Document Status Final, 2005.
53. Workflow Management Coalition Members, Workflow Client API Specification (WAPI). Technical report WfMC-TC-1002, WfMC, July, 1998. http://www.wfmc.org/
54. Workflow Management Coalition Members, Workflow Audit Data Specifications. Technical report WfMC-TC-1015, WfMC, September, 1998. http://www.wfmc.org/
55. Workflow Management Coalition Members, Terminology & Glosary. Technical report WfMC-TC-1011, WfMC, February, 1999. http://www.wfmc.org/
56. Workflow Management Coalition Members, Workflow Standard-Interoperability. Abstract Specification. Technical report WFMC-TC-102, WfMC, December, 1999. http://www.wfmc.org/
57. World Wide Web Consortium. W3C web services standard: http://www.w3.org/2002/ws/
58. M.C. Penadés. *Una Aproximación Metodológica al Desarrollo de Flujos de Trabajo*. Tesis doctoral. 2002.
59. J.Fernández. *Representación, Interpretación y Aprendizaje de Flujos de Trabajo basado en Actividades para la estandarización de Vías Clínicas*. Tesis doctoral. 2009
60. Workflow Management Coalition. *http://www.wfmc.org*.
61. Workflow Management Coalition Members, Workflow Standard-Interoperability. Abstract Specification. Technical report WfMC-TC-1012, WfMC, December, 1999. http://www.wfmc.org/
62. FlowMind. http://www.flowmind.org

63. Talmia. http://www.talmia.com

64. Workflow for ICM. http://www.infoviews.com.mx/ICM/Modulos/Workflow/

65. Process Maker. http://www.processmaker.com/

66. Lotus Workflow. http://www-142.ibm.com/software/products/es/es/workflow/

67. Microsoft Team System http://msdn.microsoft.com/es-es/library/fda2bad5(VS.80).aspx

Anexo 1. Patrones Workflow

Aquí se presenta una descripción revisada del los veinte patrones de flujo presentado previamente en [48].

A.1. Patrones básicos de control de flujo

Esta clase de patrones de captura aspectos elementales de control de proceso y son similares a las definiciones de estos conceptos propuestos inicialmente por la Workflow Management Coalition (WfMC) [55].

1. Secuencia

El patrón de Secuencia describe el proceso más simple dentro de un flujo de control. Este patrón expresa el proceso en el que una actividad se inicia después de haberse completado de otra en el mismo proceso.

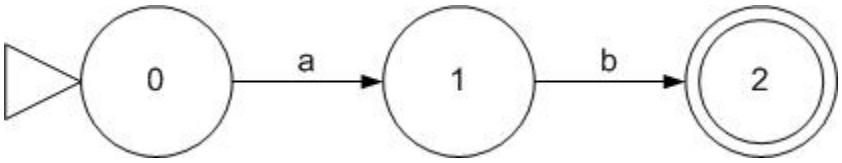


Figura 5. Secuencia

En la Figura 111 se puede ver un ejemplo de cómo se define este patrón gráficamente.

2. Separación Paralela

El patrón de separación paralela describe un proceso en el que desde una actividad completada, se da lugar a la ejecución de dos o más actividades en paralelo.

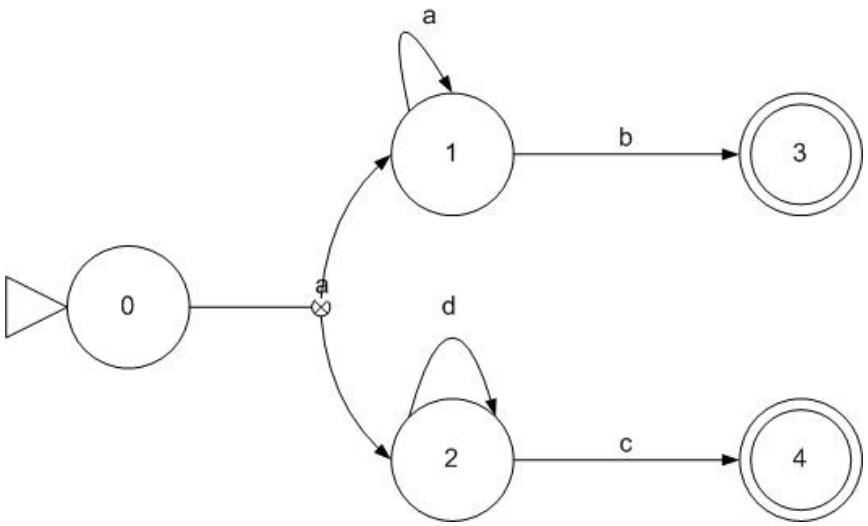


Figura 16. Separación Paralela

En la Figura 112 se puede observar un ejemplo del patrón. En esta figura, desde el estado inicial 0 se pasa, utilizando el símbolo a, a los estados 1 y 2 concurrentemente.

3. Sincronización

El patrón se sincronización viene asociado a procesos que tienen en ejecución múltiples actividades ejecutadas en paralelo. En algunos casos, es necesario que un subconjunto de las actividades que se están ejecutando en paralelo termine antes seguir ejecutando el WF. El elemento sincronizador, esperará a que todas las actividades terminen y continuará el proceso con la siguiente o las siguientes actividades.

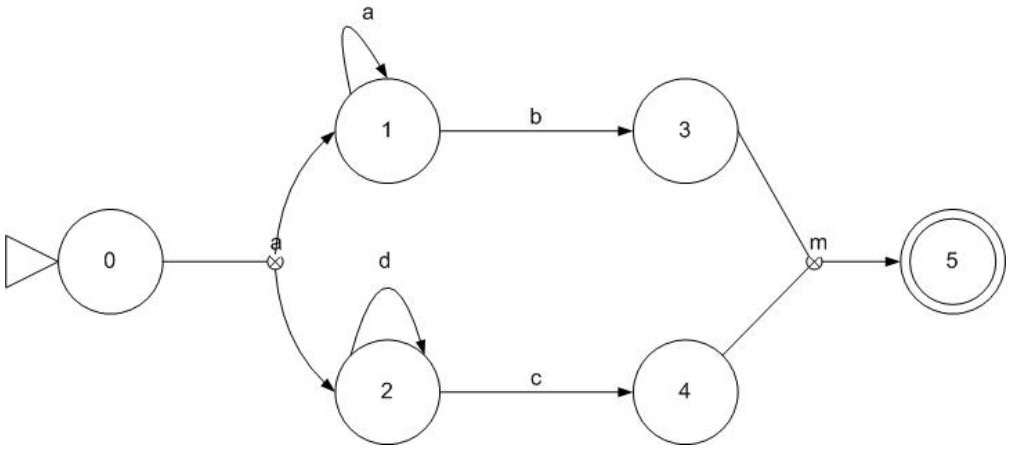


Figura 7. Sincronización

En la Figura 113 se puede ver un ejemplo de representación del patrón de sincronización. En el gráfico se puede ver un proceso de separación paralela que divide el proceso inicial. El patrón de sincronización juntará las dos secuencias de actividades en una.

4. Elección exclusiva

El patrón de Elección exclusiva permite a los workflows realizar decisiones a la hora de ejecutar actividades en función de las ramas escogidas.

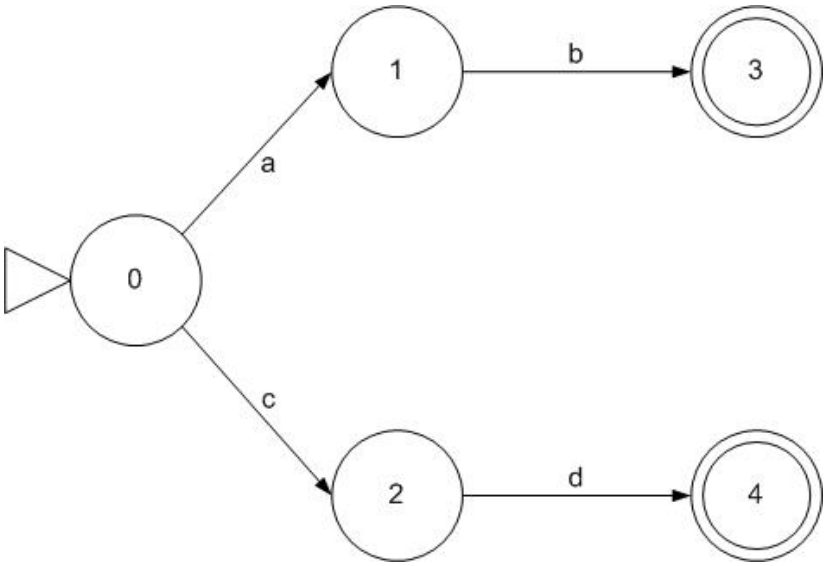


Figura 8. Elección Exclusiva

En la Figura 114 se puede ver como se define el patrón de elección exclusiva con un TPA en un ejemplo. El paso desde el estado 0 al estado 1 se produce cuando llegue un símbolo a, mientras que el paso del estado 0 al 2 se produce cuando llega un símbolo c. Estos símbolos pueden generados automáticamente por un motor al finalizar la actividad del estado 0 o pueden ejecutarse por un evento externo. En este caso, este patrón define un paso automático, el patrón que define el mismo caso de manera externa será abordado más adelante.

5. Fusión Simple

El patrón de fusión simple viene asociado a procesos donde hay caminos alternativos de ejecución que confluyen en un estado común. La separación de los caminos se puede definir usando el patrón de elección exclusiva, mientras que la fusión de los caminos en la misma rama se realiza utilizando el patrón de fusión simple. Este patrón supone que ninguna de las ramas que junta puede ser ejecutada en paralelo, estos casos serán abordados en otros patrones.

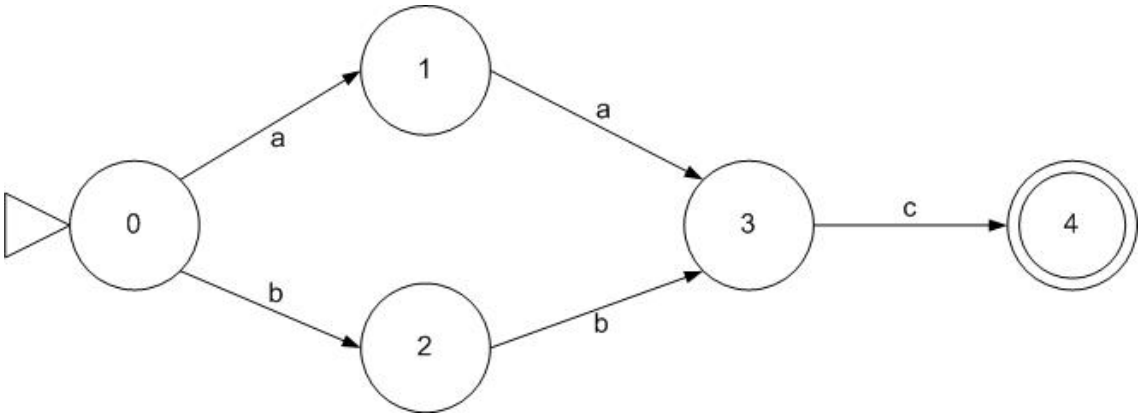


Figura 9. Fusión Simple

En la Figura 115 se define un ejemplo patrón de fusión simple. En este caso se produce una elección exclusiva de dos ramas que confluyen de los estados 1 y 2 al estado 3.

A.2. Patrones de Ramificación Avanzada y Sincronización

Aunque el conjunto de patrones de ramificación avanzada y sincronización no pertenecen al conjunto de patrones básicos por su complejidad, esto no significa que su uso sea minoritario. Este conjunto de cuatro patrones es bastante común en la definición de procesos de la vida real. En esta sección se procederá a su definición.

6. Multielección

El patrón de multielección propone una mejora del patrón de elección exclusiva. Mientras que el patrón de elección exclusiva sólo permitía ejecutar una secuencia de actividades a la vez, el patrón de multielección permite la ejecución de un número indeterminado de secuencias dependiendo de la selección en tiempo de ejecución.

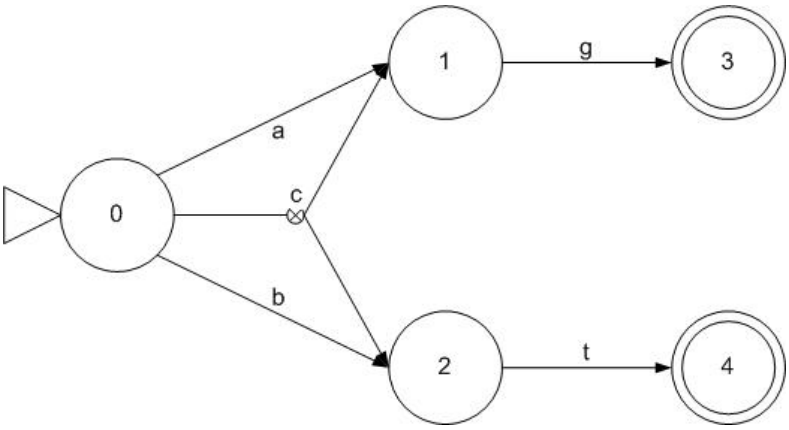


Figura 10. Multielección

En la Figura 116 se presenta un ejemplo de patrón de multielección. El paso del estado 0 a los estados 1 y 2 está regulado por tres arcos, uno, marcado con el símbolo a que

lleva al estado 1, otro, marcado con el símbolo b que lleva al estado 2, y un tercero, marcado con el símbolo c que lleva a la ejecución en paralelo de ambos estados. Cada arco marca una posibilidad de ejecución del proceso, marcado por el símbolo que utilicemos.

7. Fusión Sincronizada

El patrón de función sincronizada funciona de manera análoga al patrón de multielección, sólo que al final de un conjunto de secuencias de actividades. Cuando en un proceso se ejecuta un patrón de multielección, es más que probable que en algún momento el conjunto de las secuencias de actividades potencialmente seleccionables para ser ejecutadas confluyan en un mismo punto. Es este caso el principal problema es saber cómo han de continuar, es decir, que estados hay que fusionar. El patrón de fusión sincronizada define el proceso de fusión de varios hilos de secuencias de actividades en uno, sincronizándolas según se hayan ejecutado.

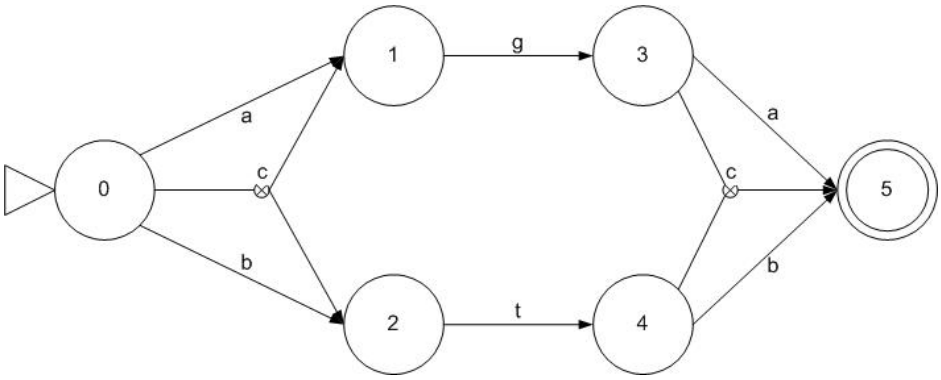


Figura 11. Fusión Sincronizada

En la Figura 117 se muestra un ejemplo del patrón de fusión sincronizada. En el ejemplo podemos ver como el proceso separa la secuencia inicial en dos secuencias de actividades conforme a los símbolos recibidos y, posteriormente, los vuelve a juntar correspondientemente a como hayan sido ejecutados.

8. Multifusión

El patrón de multifusión es una alternativa funcional al patrón de fusión sincronizada. Mientras que el patrón de fusión sincronizada, exige que las secuencias de actividades que fueron seleccionadas inicialmente deban de terminar sincronizadamente, el patrón de multifusión permite a las secuencias fusionarse sin espera y continuar así el resto del proceso.

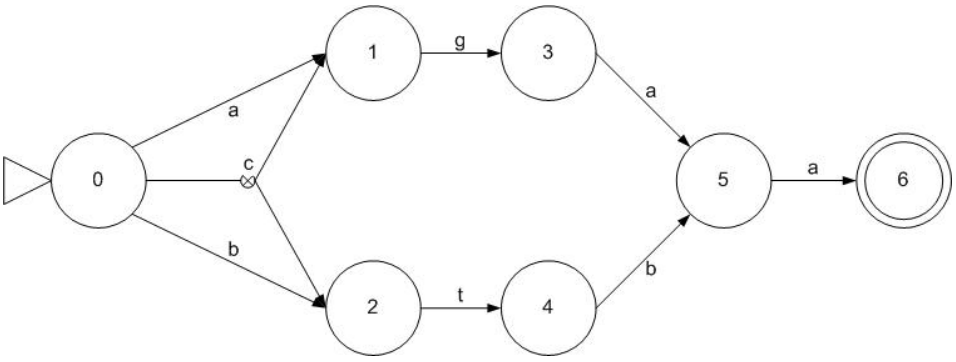


Figura 12. Multifusión

En la Figura 118 se muestra un ejemplo de fusión múltiple. En el ejemplo, la secuencia inicial es separada en hasta dos secuencias de actividades y a continuación son fusionadas por separado sin arcos que exijan que los estados de los extremos finales de las secuencias (3 y 4) deban de estar activos.

9. Discriminador

Cuando se aplica un patrón de fusión múltiple que ejecuta más de una secuencia de actividades a la vez, las actividades que se ejecutan después del patrón, lo hacen más de una vez y esto no es siempre deseable. La alternativa más obvia a este problema es la fusión sincronizada, sin embargo, no siempre es deseable a esperar a que se acaben las actividades para continuar el proceso. De esta necesidad surge el patrón de discriminación. Este patrón, deja continuar la primera de las secuencias de actividades que lleguen a él ignorando el resto de las secuencias. De este modo se solucionan los problemas de la duplicidad de ejecución de actividades.

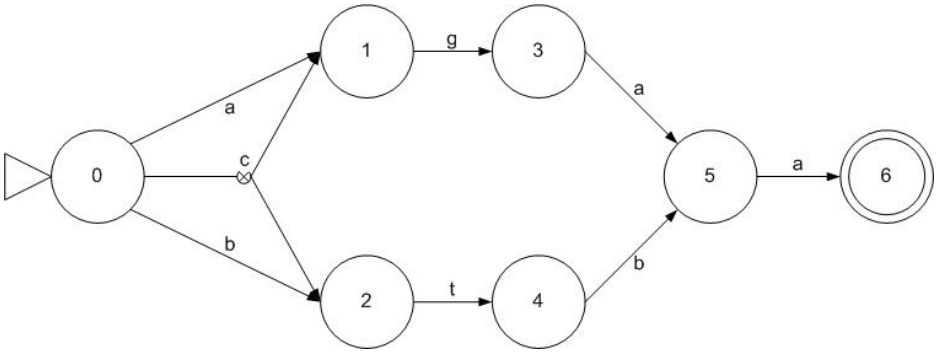


Figura 13. Discriminador

En la Figura 119 se observa el ejemplo del patrón de discriminación.

A.3. Patrones Estructurales

Algunos sistemas de gestión de WF imponen restricciones en sus modelos de WF, como por ejemplo impedir los ciclos arbitrarios en las definiciones de WF. Estas restricciones no son siempre naturales, y tienden a restringir la libertad de especificación de los diseñadores de WF. En esta sección se presentan dos patrones que representan típicas restricciones estructurales de los sistemas de gestión de WF y que, no obstante, suelen ser interesantes para la definición de estos.

10. Ciclos Arbitrarios

Un ciclo arbitrario es un punto donde un conjunto de una o más actividades pueden ser ejecutadas repetidamente.

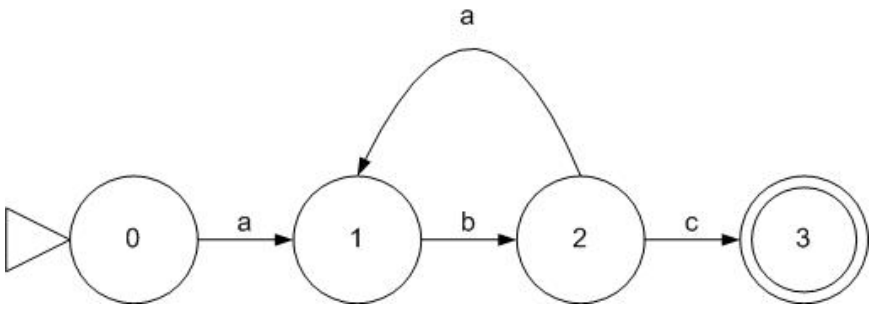


Figura 114. Ciclo Arbitrario

En la Figura 120 podemos ver un patrón de ciclo arbitrario.

11. Terminación Implícita

En algunos motores de WF un proceso sólo termina cuando se ha llegado a un estado final. Esto no es siempre deseable a nivel de implementación, ya que en determinados momentos, procesos que no llegan a estados finales deben terminar. El patrón de terminación implícita dice que un proceso debe terminar, no solo cuando llegue a un estado final, sino cuando ya no tenga ninguna actividad que hacer.

A.4. Patrones que involucran Múltiples instancias

Desde un punto de vista teórico, los patrones de múltiples instancias corresponden a múltiples hilos de ejecución que se refieren a la misma definición compartida. Sin embargo, desde un punto de vista práctico, su implementación no es tan sencilla debido sobre todo a restricciones de diseño.

Cuando hablamos de patrones que involucran múltiples instancias, consideramos dos tipos de habilidades: habilidades para lanzar múltiples instancias de una actividad o

subproceso, o habilidades para sincronizar estas actividades. Todos los patrones de esta sección cumplen el primer patrón, sin embargo, las diferencias entre los cuatro patrones de esta sección versan en la capacidad de la sincronización de estas actividades o subprocesos.

12. Múltiples instancias Sin Sincronización

El patrón de múltiples instancias sin sincronización define que una actividad es capaz de ejecutar instancias diferentes de la misma actividad. Estas instancias son independientes de las instancias de otros hilos, además de que no se requiere de sincronización entre ellas.

13. Múltiples instancias con conocimiento en tiempo de Diseño

El patrón de múltiples instancias con conocimiento en tiempo de diseño define que una actividad es capaz de ejecutar instancias diferentes de la misma actividad en un número conocido en tiempo de diseño del WF. Solo cuando todas las actividades se completasen las siguientes actividades se iniciarían.

14. Múltiples instancias con conocimiento en tiempo de Ejecución

El patrón de múltiples instancias con conocimiento en tiempo de ejecución define que una actividad es capaz de ejecutar instancias diferentes de la misma actividad en un número conocido en tiempo de ejecución del WF. Solo cuando todas las actividades se completasen las siguientes actividades se iniciarían.

15. Múltiples instancias sin conocimiento previo

El patrón de múltiples instancias sin conocimiento previo define que una actividad es capaz de ejecutar instancias diferentes de la misma actividad en un número desconocido a priori. Solo cuando todas las actividades se completasen las siguientes actividades se iniciarían.

A.5. Patrones Basados en Estados

En los flujos de trabajo de la vida real, las instancias normalmente se encuentran en un estado esperando a ser procesadas. Los patrones basados en estados son tres patrones en los que los procesos pueden estar esperando indefinidamente en un estado dependiendo de factores externos o internos al proceso.

16. Elección Derivada

El patrón de Elección Derivada define el proceso por el cual desde una actividad completada se pasa a un estado u otro dependiendo de la opción elegida. Este patrón se

diferencia de los anteriores patrones de elección en que dicha elección la realiza un agente externo. El WF iniciado se queda en un estado suspendido hasta que el agente externo da la señal para la continuación del WF.

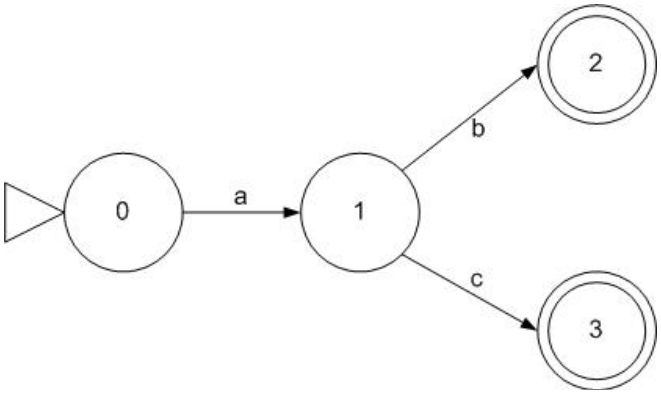


Figura 115. Elección Derivada

En la Figura 121 se puede ver un ejemplo de patrón de elección derivada. En este gráfico se ve cómo se puede pasar del estado 1 al 2 con una b o del 1 al 3 con una c.

17. Rutina paralela entrelazada

El patrón de rutina paralela entrelazada define un proceso por el cual un conjunto de secuencias de actividades deben ejecutarse en un orden arbitrario, pero nunca dos actividades pueden ejecutarse al mismo tiempo. Si una de las secuencias de actividades empieza el resto han de permanecer en un estado de espera a que termine la primera para continuar.

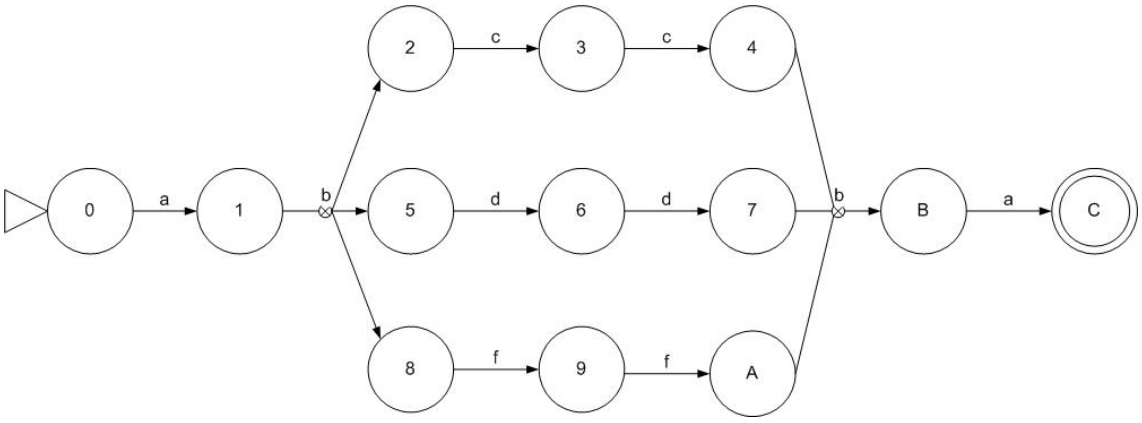


Figura 116. Rutina paralela entrelazada

En la Figura 122 se puede ver un ejemplo de rutina paralela entrelazada. Existen tres secuencias de actividades correspondientes a los estados 2-3-4, 5-6-7 y 8-9-A que se han de ejecutar de manera arbitraria, pero sin ejecutarse al mismo tiempo. Los estados

iniciales de las secuencias 2, 5 y 8 son estados de espera donde las actividades esperan a ser ejecutadas mientras, que los estados 4, 7 y A son estados finales de espera de cada una de las secuencias de actividades, donde una vez ejecutadas, estas esperan a que las demás secuencias terminen.

18. Hito

El patrón de Hito se basa en la idea en que la habilitación de una actividad de una secuencia de actividades depende de si un estado en una secuencia de actividades paralela ha sido alcanzado o no. En el caso en que el estado 'hito' no haya sido alcanzado, y la secuencia no pueda continuar, la secuencia esperará indefinidamente hasta que el estado haya sido alcanzado.

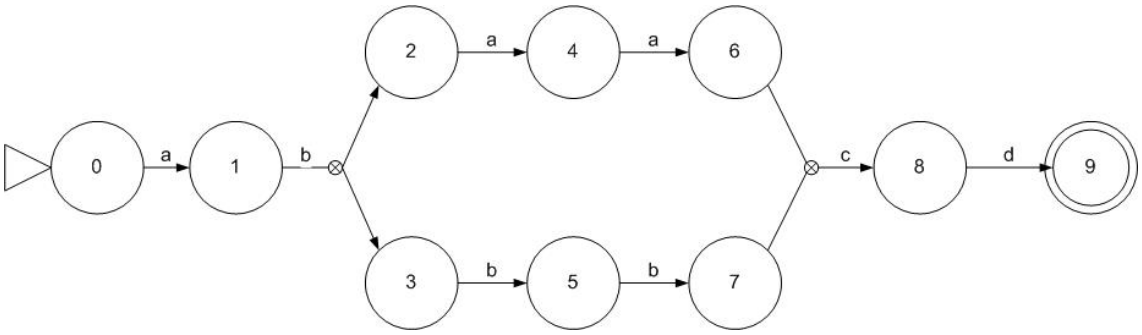


Figura 117. Hito

En la Figura 123 se puede observar un ejemplo de patrón de hito. En este ejemplo se toma que el estado 4 es un hito para la secuencia 3-5-7 y el estado 5 es un hito para la secuencia 2-4-6.

A.6. Patrones de Cancelación

Los patrones de cancelación son referidos a la posibilidad de cancelar actividades o instancias en tiempo de ejecución.

Como en otros patrones, este tipo de patrones es totalmente dependiente del motor, con lo que no existe ninguna diferencia a nivel formal.

19. Cancelar Actividad

El patrón de cancelar actividad se refiere al caso en el que la ejecución de una actividad es deshabilitada

20. Cancelar Instancia

El patrón de cancelar Instancia se refiere al caso en el que la ejecución una instancia de un WF es deshabilitado

Anexo 2. Ejemplo de workflows implantados

A continuación se van a añadir capturas de workflows implantados en la PYME con la que hemos colaborado durante más de tres años.

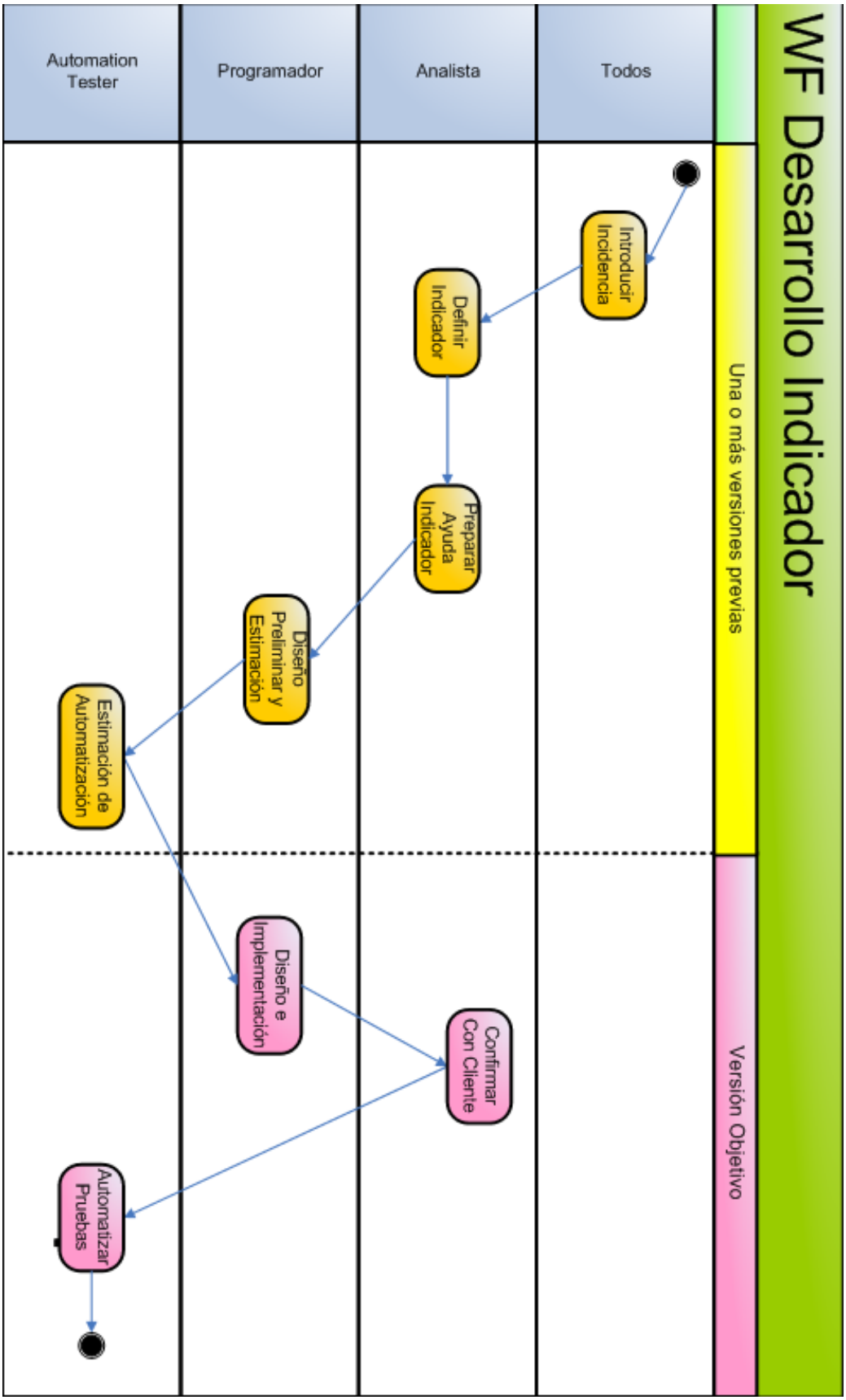


Figura 124. WF Desarrollo Indicador

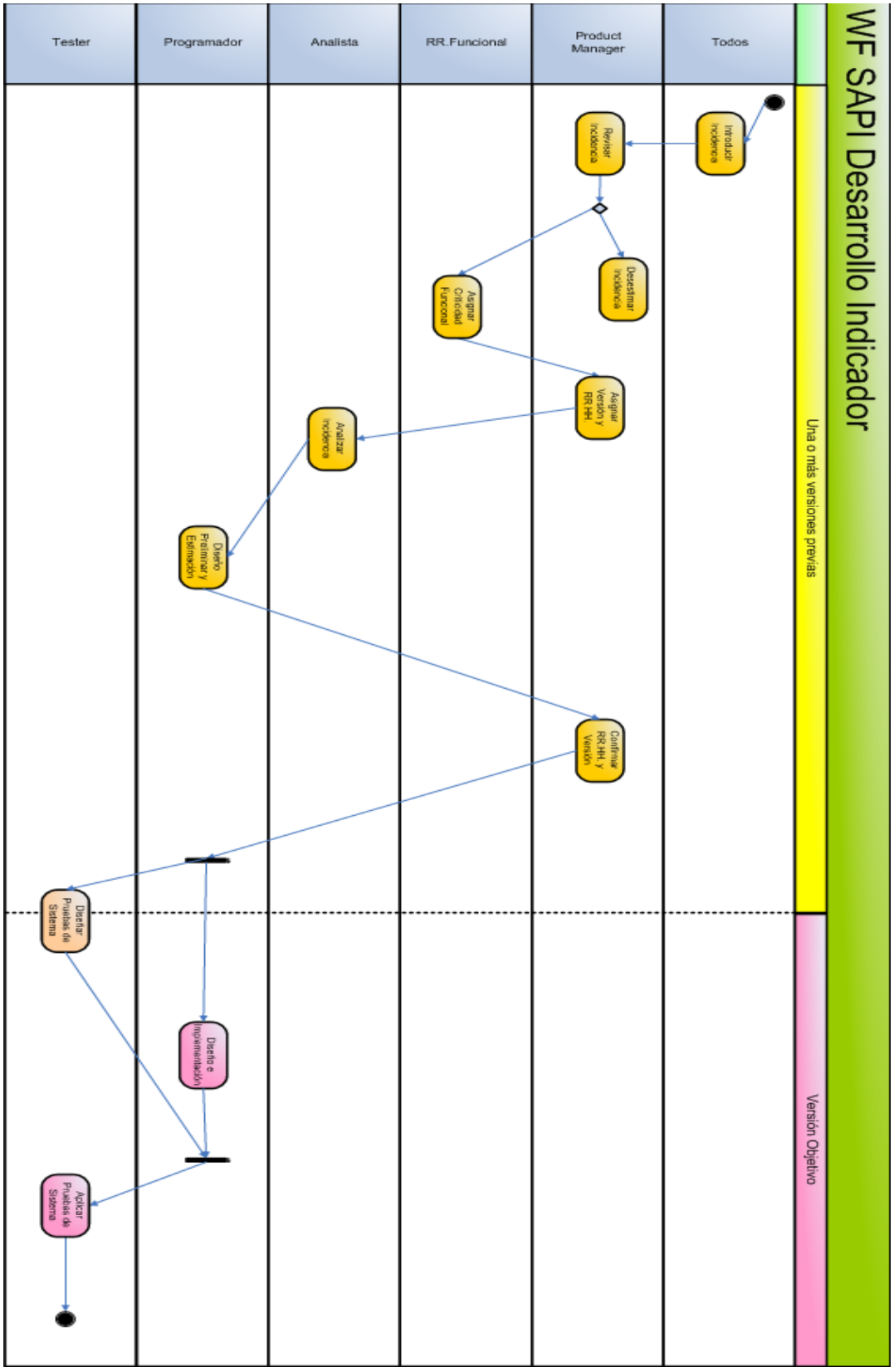


Figura 125. WF SAPI Desarrollo de Indicador

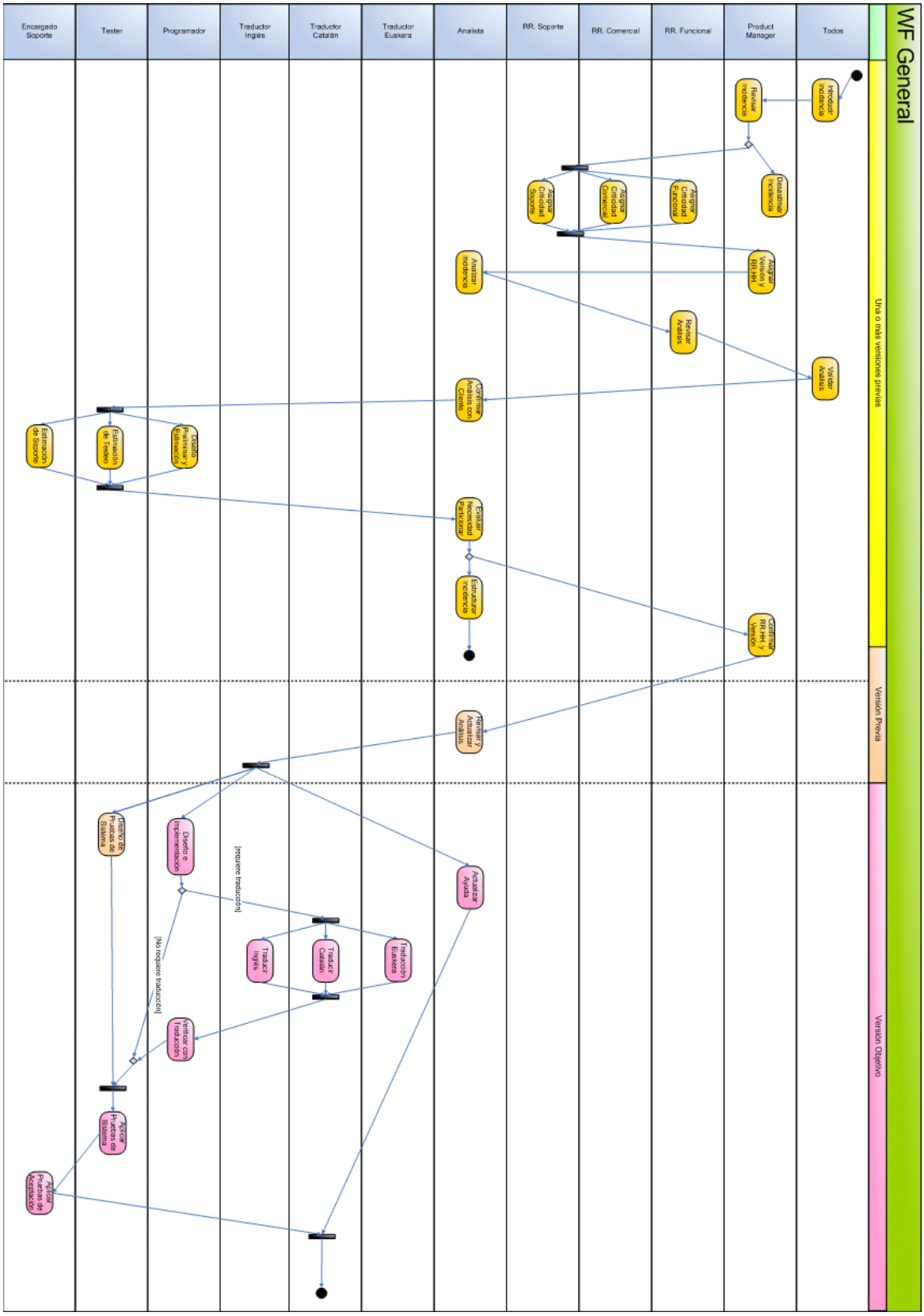


Figura 126. WF General