Un Entorno para el Desarrollo de Modelos de Flujo de Trabajo[1]

Mª Carmen Penadés, José H. Canós

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
Camí de Vera, s/n
46022 València (Spain)
{mpenades, jhcanos}@dsic.upv.es

Resumen. Los sistemas de gestión de flujos de trabajo permiten abordar la automatización de los procesos que tienen lugar dentro de las organizaciones. El desarrollo de estos sistemas ha tenido un fuerte componente tecnológico, centrándose más en la creación de entornos de ejecución eficientes y distribuidos, que en proporcionar un soporte formal y metodológico que garantice la obtención de modelos de flujo de trabajo de calidad. En este trabajo se presenta un entorno para el desarrollo de modelos de flujo de trabajo. Un editor permite la construcción de dichos modelos utilizando el lenguaje gráfico definido en base a un metamodelo de referencia. A partir de este modelo, almacenado en un repositorio, se genera automáticamente una especificación formal equivalente. Esta especificación es ya un prototipo ejecutable que sirve para mejorar y validar dicho modelo, corrigiendo posibles errores antes de su ejecución. Tras la validación, generadores de código producen automáticamente versiones del modelo en entornos eficientes.

1 Introducción

Aunque existen en el mercado gran número de productos denominados genéricamente sistemas de gestión de flujos de trabajo (SGFT) o *workflow management systems*, como por ejemplo [14], [13] o [1]; no todos ellos entran dentro de la misma categoría (producción, colaboración, administrativo o ad-hoc) ni proporcionan las mismas prestaciones [3], [12]. Sin embargo, y de acuerdo con la *Workflow Management Coalition* (WfMC), todos ellos se caracterizan por dar soporte a tres áreas funcionales [11] (ver figura 1):

- La funcionalidad de definición o construcción del flujo de trabajo (FT), denominada *built-time functions*, tiene que ver con las facilidades que ofrecen los SGFT de definir, modelar y especificar FT de acuerdo con un metamodelo ya establecido y que manejará los típicos conceptos de actividad, actor, datos, aplicaciones invocadas, etc.

[1] Este trabajo ha sido parcialmente subvencionado por la CICYT. Proyecto de investigación DOLMEN-SIGLO TIC2000-1673-C06-01.

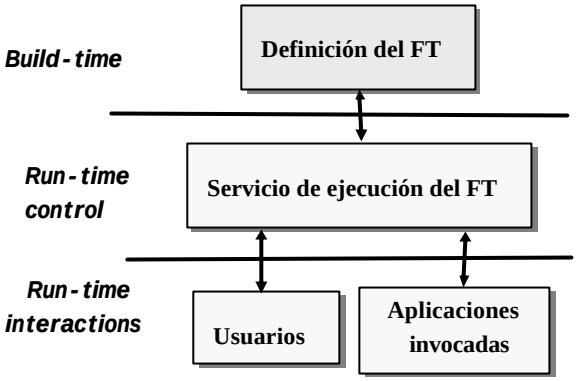

Figura 1. Funcionalidad común de los Sistemas de Gestión de Flujos de Trabajo.

- Las funciones de control de ejecución (*run-time control functions*) permiten generar una versión ejecutable del FT especficado y que ésta pueda ser ejecutada sobre una máquina. Para ello, se creará la correspondiente instancia del FT y se controlará la creación y ejecución de las instancias actividad de acuerdo con el flujo de control preestablecido.
- El área denominada *run-time interactions* agrupa toda la funcionalidad que proporciona un SGFT para que sea posible la comunicación con usuarios, en aquellas actividades que requieran participación humana, así como la posibilidad de invocar aplicaciones externas para la ejecución de ciertas actividades. Igualmente, permiten acceder y/o almacenar los datos de entrada y/o salida de las actividades, los cuales normalmente suelen estar en una base de datos.

Puesto que la finalidad última de un SGFT es la automatización de un proceso de negocio, a través del FT diseñado, los principales esfuerzos se han centrado en las funcionalidades de *runtime*, es decir, de ejecución en entornos eficientes y en ocasiones distribuidos. Por el contrario, para la construcción del modelo de FT que posteriormente se va a ejecutar, siguen existiendo limitaciones importantes [9], [20].

A pesar de los esfuerzos realizados para la construcción de estándares en el campo de los SGFT, por parte de la WfMC, no existe un metamodelo de FT aceptado e implementado por la mayor parte de las herramientas existentes en el mercado. Esta falta de formalización en el metamodelo incide de forma importante en las prestaciones que se ofrecen a nivel de construcción o definición del FT. En la mayoría de las herramientas se proporciona un editor gráfico para su definición, sin tener clara cuál es la semántica asociada al modelo que se construye, puesto que no existe una formalización del mismo, siendo, en ocasiones, muy dependiente de las propias facilidades de ejecución que se proporcionen. Todo ello incide en otros problemas importantes de los SGFT, como son la falta de compatibilidad entre diferentes SGFT (lo cual dificulta enormemente la interoperabilidad) o la poca escalabilidad de los mismos [2].

En este trabajo se presenta un entorno orientado a la construcción de modelos de FT. Este entorno permite modelar de forma gráfica un FT, de acuerdo a un metamodelo formalmente definido, el cual es almacenado en un reporsitorio. Además, permite utilizar la técnica del prototipado para validar dicho modelo. Para ello, se genera una especificación equivalente en el lenguaje de especificación formal y orientado a objetos OASIS [15] [7]. La semántica operacional del lenguaje OASIS nos garantiza el soporte a la validación. Esta especificación puede ser animada posteriormente en un entorno de ejecución de OASIS, teniendo así un prototipo ejecutable con el que el analista puede validar los requisitos del proceso e introducir mejoras en el mismo, antes de su ejecución en un entorno eficiente. Una vez construido el modelo y validado, el entorno generaría especificaciones equivalentes en los distintos lenguajes de ejecución de los sistemas a los que se quiera exportar el modelo para su ejecución, como por ejemplo OPERA [10] o MQSeries WF [14].

La estructura de este trabajo es la que sigue. En el apartado 2 se muestra una visión global del entorno desarrollado, citándose sus distintos componentes. En el apartado 3 se describe el lenguaje gráfico definido para el modelado de FT, pasando en el apartado 4 a enumerar las principales características del editor gráfico e introducir un ejemplo. En el apartado 5 se resumen los patrones de traducción para la obtención automáticamente de la especificación OASIS que representa el FT modelado. Finalmente, aparecen las conclusiones y trabajos futuros.

2 Visión Global del Entorno

La figura 2 muestra los distintos componentes que forman este entorno. El analista puede utilizar un lenguaje gráfico para construir el modelo de FT que represente el proceso de negocio seguido por la organización. Este lenguaje se ha definido en base a un metamodelo de FT establecido y permite que el analista no tenga que conocer en detalle la sintaxis del lenguaje de ejecución que se utilice. En el apartado 3 se describe el metamodelo y el lenguaje gráfico con mayor nivel de detalle. Toda la información del modelo gráfico construido queda almacenada en un repositorio relacional, cuya estructura está basada en el metamodelo de referencia.

Sin embargo, en muchas ocasiones, especialmente cuando el proceso de negocio es complejo y no totalmente conocido, el analista necesita descubrir dicho proceso antes de modelarlo. Para ello, se ha incluido en el entorno una capa de ingeniería de requisitos que permite ayudar al analista a capturar los requisitos del proceso de negocio, expresados mediante casos de uso; a partir del modelo de casos de uso, se obtiene automáticamente una versión preliminar del FT de la organización. Este modelo de FT se puede refinar desde el editor hasta completar el modelo definitivo. El conversor del modelo de casos de uso al modelo de FT está basado en las equivalencias entre ambos modelos, junto con una serie de patrones de proceso definidos a partir de las relaciones de uso y extensión entre casos de uso. Una descripción más detallada del módulo de conversión puede consultarse en [8] y [18].

Una vez construido el modelo de FT, el entorno permite animar el modelo obtenido con la finalidad de iniciar un proceso de validación del mismo. Para ello, el generador de código OASIS permite, a partir del modelo construido, generar automáticamente la

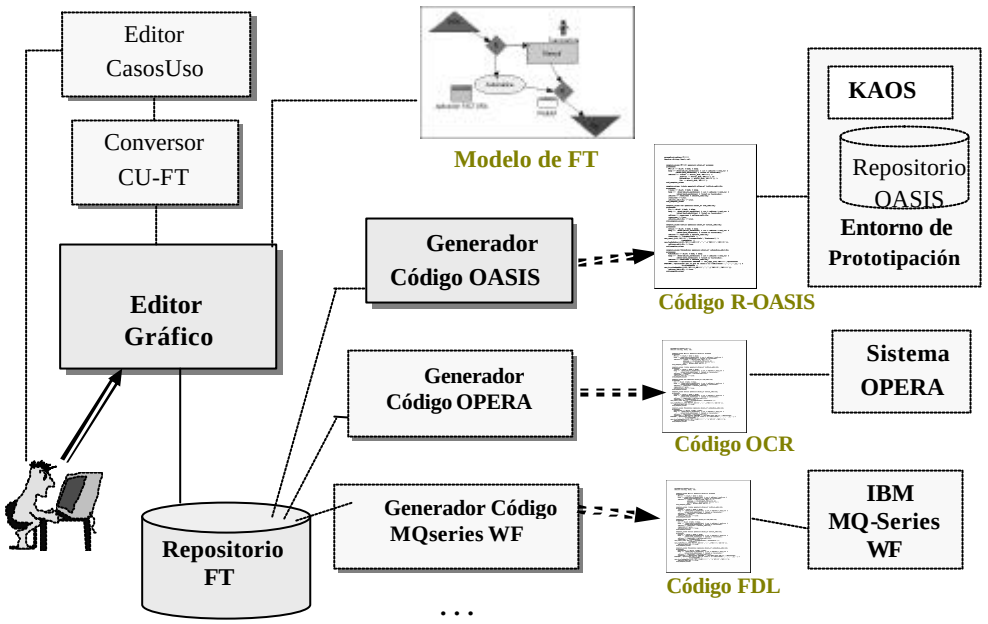

Figura 2. Visión global del entorno

especificación formal equivalente, que puede ejecutarse en un entorno de animación de OASIS, como es el caso de KAOS [6]. Este generador está totalmente implementado y en el apartado 5 se resumen las principales pautas de traducción que se aplican.

Del mismo modo, pueden existir otros generadores de código que permitan obtener una versión ejecutable del modelo de FT en diferentes sistemas de ejecución de FT. En concreto, este entorno va a dar soporte a que un modelo de FT se pueda ejecutar en el sistema OPERA y en el sistema MQ-Series WF. La implementación de estos módulos de generación está actualmente en desarrollo, a partir de las equivalencias establecidas entre nuestro metamodelo y el que soportan cada uno de estos sistemas.

3 Un Lenguaje Gráfico para el Modelado de Flujos de Trabajo

Se ha definido un lenguaje gráfico que permite captar las distintas dimensiones de un FT. Este lenguaje está basado en el metamodelo de FT que se muestra en la figura 3[2], en notación UML [5]. Dicho metamodelo se ha formalizado en OASIS, lo que proporciona una semántica formal. En [16] se presenta la formalización del concepto

[2] Las clases y relaciones que forman el metamodelo aparecen denotadas por términos en inglés, siendo éstos los utilizados en la formalización del mismo y en la implementación de los generadores correspondientes.

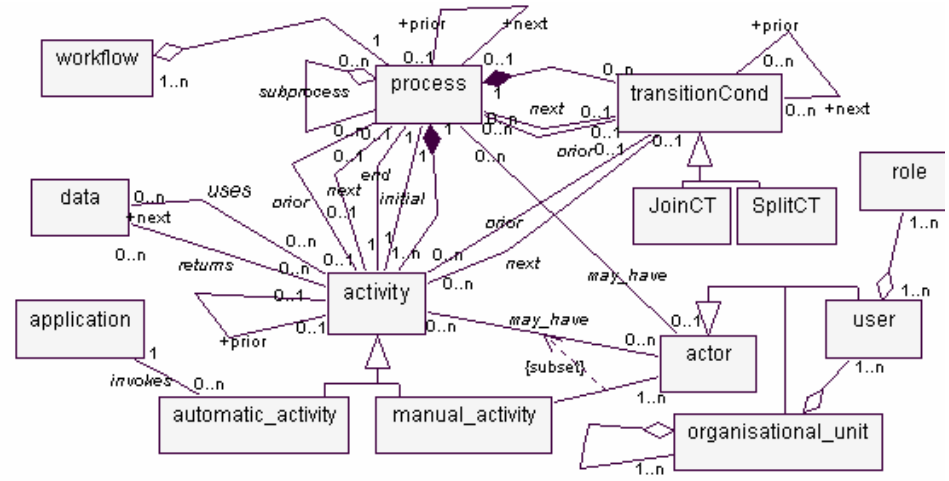

Figura 3. Metamodelo de flujos de trabajo

de proceso en OASIS y en [17] se encuentra la especificación OASIS completa del metamodelo.

El modelo que se construye con el lenguaje gráfico definido está formado por el proceso, como entidad fundamental, y los recursos asociados al mismo. El proceso es un conjunto de entidades relacionadas mediante flujos de control, donde las entidades pueden ser actividades o subprocesos o condiciones de transición. Los recursos incluyen tanto a los actores del proceso, como a los datos y aplicaciones que se utilicen. La notación gráfica aparece en la figura 4. A continuación se describen las principales características de cada uno de éstos elementos gráficos, así como las propiedades de los mismos, de acuerdo con el metamodelo definido.

- **Subprocesos**: Las actividades que forman parte del FT pueden ser complejas y representar en sí mismas un proceso. En este caso, se pueden modelar subprocesos como parte de un proceso. Esto introduce modularidad en la definición del FT, facilitando el modelado de sistemas complejos. Las propiedades de un proceso son: nombre, descripción, condición de inicio y condición de finalización.
- **Actividades**: Son los pasos básicos a realizar dentro del proceso. Sus propiedades son: nombre, descripción, condición de inicio, de finalización y acciones concretas asociadas. Existen distintos tipos de actividades: actividades manuales son las que obligatoriamente requieren de la participación humana para su realización; actividades automáticas son tienen como acción asociada la invocación de una aplicación externa. Además, todo proceso tiene una actividad inicial y una actividad final que marcan el inicio y final del mismo, respectivamente.
- **Condiciones**: Pueden ser de unión o de bifurcación, distinguiéndose varios casos.
  Las ***condiciones de unión*** son las que recogen varios flujos de entrada y tienen un único flujo de salida. Los flujos de entrada pueden tener asociadas condiciones. Se distingue entre:

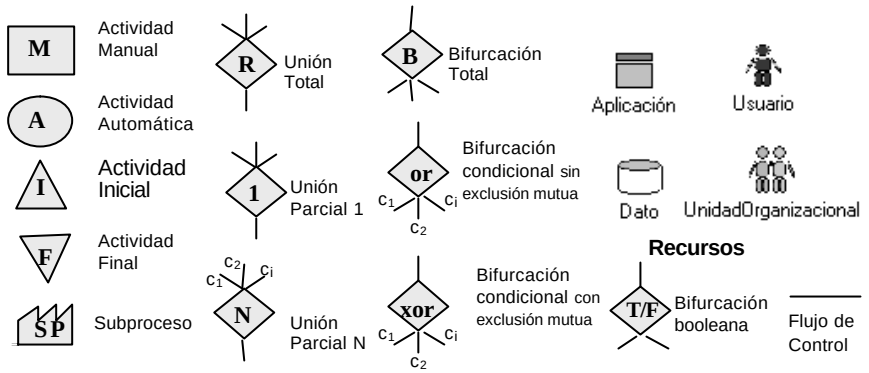

Figura 4. Notación gráfica para la definición de un FT

- *Unión total* o *punto de reunión*. Esta condición expresa que hasta que el control no llegue a todos los flujos de entrada, éste no pasa al flujo de salida. No tiene condiciones asociadas.
- *Unión parcial*. En este caso, no es necesario que el control llegue a todos los flujos de entrada, de forma que: *a) Parcial 1.* La llegada del primer flujo de entrada hace que el control pase al flujo de salida, sin tener en cuenta lo que ocurra en el resto de flujos de entrada. *b) Parcial N*. Los flujos de entrada pueden tener condiciones asociadas, entonces se espera que el control llegue a un número determinado (N; N>=1) de flujos de entrada que además cumplan la condición asociada, para que éste pase al flujo de salida.

Las ***condiciones de bifurcación*** son las que tienen un único flujo de control de entrada y varios de salida. En este caso, los flujos de salida pueden tener asociadas condiciones. Se distingue entre:

- *Bifurcación total* o *punto de bifurcación*. Esta condición expresa que cuando el control llegue al flujo de entrada, éste se encamina hacia todos los flujos de salida, para que se inicie su ejecución en paralelo. No tiene condiciones asociadas.
- *Bifurcación condicional*. Cuando llega el control al flujo de entrada, se evalúan las condiciones asociadas a los flujos de salida, para determinar a cuáles de ellos va a pasar el control y continuar con la ejecución. Se distinguen dos casos: *a) Sin exclusión mutua*. Las condiciones que etiquetan los flujos de salida no son mutuamente excluyentes, por tanto, el control pasará a todos aquellos flujos de salida cuya condición asociada sea cierta. *b) Con exclusión mutua*. Puesto que las condiciones asociadas a los flujos de salida son mutuamente excluyentes, sólo una de ellas será cierta. Esto determina a qué flujo de salida pasa el control.
- *Bifurcación booleana*. Es un caso particular de la bifurcación condicional con exclusión mutua, puesto que tiene dos flujos de salida y la misma fórmula asociada para ambos, pero evaluada a cierta en uno y a falsa en el otro. Puesto que este tipo de condición suele ser frecuente, se ha incluido la facilidad de poder representarla directamente.

- **Flujo de control**: Conecta las entidades, describiendo la secuencia de ejecución o evaluación de las entidades que forman el proceso. Un flujo de control siempre

tiene una entidad origen y otra destino. Las actividades inicial y final, puesto que delimitan el inicio y final de un proceso, sólo admiten un flujo de salida o uno de entrada, respectivamente. El resto de actividades, al igual que los subprocesos, admiten dos flujos: uno de entrada y otro de salida, puesto que siempre van precedidos y sucedidos por una sola entidad. Las entidades que sí pueden tener varios flujos de salida o varios flujos de entrada son las condiciones de transición, según los distintos tipos comentados anteriormente.

Los recursos se pueden asociar a las entidades. Dichas asociaciones representan las diferentes relaciones que existen en el metamodelo, tales como qué aplicación es invocada por una actividad automática, qué datos son utilizados por una actividad o qué actor es responsable de un proceso. A continuación se detallan las propiedades para cada uno de ellos:

- **Aplicaciones**: nombre, descripción, camino donde se encuentra ubicada, llamada y parámetros para su invocación y permisos.
- **Datos**: nombre, descripción y camino donde se encuentra ubicado.
- **Actores**: nombre del actor y descripción. Pero, puesto que las unidades organizacionales están formadas por usuarios que pueden organizarse jerárquicamente, cuando se define una unidad se puede indicar qué usuarios forman parte de ella, así como las unidades que la componen. Se proporcionan dos elementos gráficos distintos para modelar la participación humana en un FT, pudiéndose distinguir si el actor que inicia o participa en una actividad es un usuario concreto, o bien, una unidad organizacional. En el segundo caso, lo que se está indicando es que cualquier usuario de dicha unidad organizacional puede llevar a cabo la actividad.

4 Editor Gráfico

Las principales características del editor gráfico, a nivel de modelado de FT se pueden resumir en:

- Se ha introducido el concepto de *Proyecto* para agrupar todos aquellos procesos que están relacionados entre sí. Así, un proceso se define una sola vez, pero se puede reutilizar tantas veces como se desee, como subproceso de otro proceso más general. Un proyecto siempre tiene asociado al menos un proceso. Por otro lado, para facilitar el refinamiento del modelo del FT, conforme se van perfilando los requisitos del proceso, se permiten dos acciones: convertir en subproceso una actividad y viceversa, es decir, convertir una actividad en un subproceso.
- Los recursos (actores, datos y aplicaciones) que se definen son globales, es decir, no pertenecen a ningún proyecto en concreto. Se definen una sola vez, pero se pueden asociar a varias entidades (subprocesos, actividades y condiciones de transición) dentro del mismo proyecto o de proyectos distintos. De esta forma también se facilita la reutilización de los mismos. Cuando se define un recurso, dependiendo de si se trata de un actor, un dato o una aplicación, tiene unas propiedades por defecto a las que se les puede dar un valor, como ocurre cuando se define una entidad, pero además se pueden añadir nuevas propiedades a dichos

recursos. Para la definición de las mismas se indica el nombre de la nueva propiedad, el tipo de la misma y el valor por defecto.

- Se puede realizar, en cualquier momento, una comprobación de la corrección del FT que se está definiendo. Esta comprobación se puede realizar a nivel de proceso o de proyecto. Lo que se comprueba es si el modelo cumple las restricciones del metamodelo de referencia. Así, posibles errores que se detectan son: recursos dados de alta en el proyecto como parte de algún proceso, pero no asociados a ninguna entidad; errores en el número de flujos de entrada o salida de las condiciones de transición, dependiendo del tipo de las mismas; la no existencia de la actividad inicial o final de un proyecto, etc.

4.1 Arquitectura

El editor se ha desarrollado siguiendo una arquitectura típica de tres niveles: nivel de presentación, nivel de lógica de la aplicación y nivel de persistencia. El nivel de presentación implementa una interfaz gráfica de usuario (IGU), con menús y barras de herramientas, siguiendo las guías de estilo de una aplicación Windows de interfaz de múltiple documento (MDI). La figura 5 muestra el aspecto general de la IGU. En el área de dibujo se definen los diferentes procesos que pueden formar un proyecto, proporcionándose a través de la barra de herramientas y los menús todos los elementos gráficos para modelar el FT; la introducción del valor de las propiedades asociadas a los mismos se realiza mediante sucesivos diálogos. Además, en el árbol de accesos se muestran todos los procesos abiertos del proyecto, así como sus entidades colocadas jerárquicamente.

La funcionalidad básica del nivel intermedio es la gestión en memoria de los objetos correspondientes a los diferentes componentes del FT (gráficos y no gráficos). Finalmente, el nivel de persistencia, se encarga del acceso al repositorio, independizando al resto de la aplicación de dicha tarea. Se trata de un repositorio relacional, cuyas tablas contienen toda la información del FT diseñado. Esta información hace referencia tanto al valor asignado a cada una de las propiedades de cada componente, como a información gráfica del mismo (se almacenan las coordenadas de su posición gráfica). La estructura completa del repositorio se puede consultar en [19]. Se ha utilizado *Borland Delphi Client/Server* [4] como herramienta de desarrollo y *Paradox* como sistema gestor de bases de datos.

4.1 Un Ejemplo: Agencia de Viajes

A continuación se muestra el modelo de FT construido con la herramienta para el proceso seguido por una agencia cuando gestiona la reserva de viajes. La agencia ofrece a los clientes la posibilidad de ocuparse de la gestión de sus viajes. El cliente elige el itinerario deseado, aportando sus datos personales y una tarjeta de crédito con la que realizar el pago. La agencia se encarga de realizar las reservas oportunas. Estas reservas incluyen no sólo los hoteles elegidos en las diferentes ciudades, sino también la reserva del medio de transporte elegido y la posibilidad de alquilar un vehículo para utilizar durante la estancia. Una vez la agencia tiene confirmadas las reservas de todas

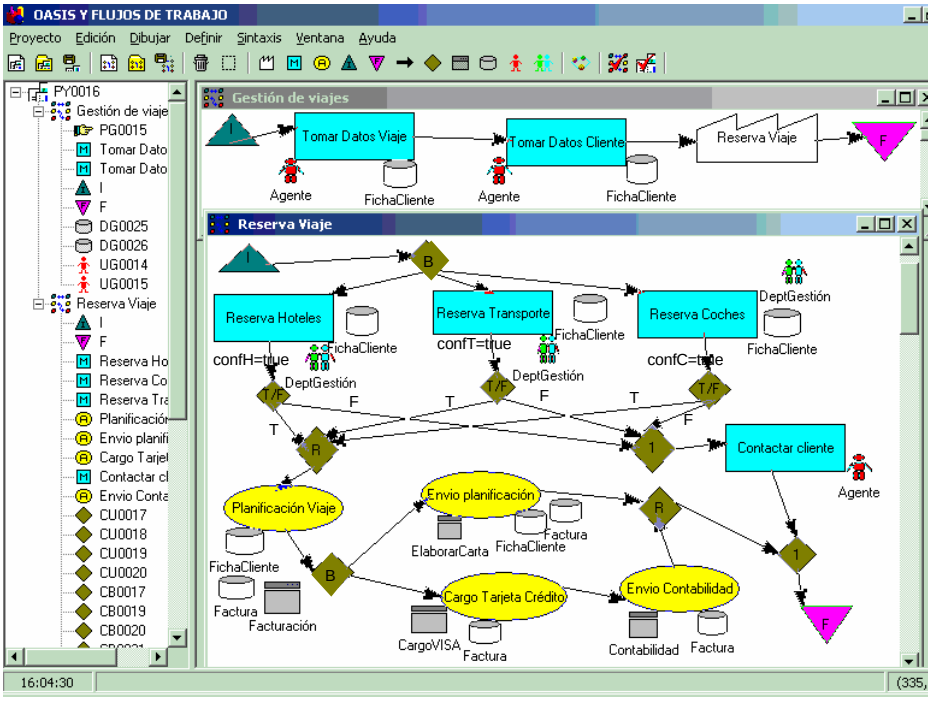

Figura 5. Ejemplo de modelado con la herramienta

las peticiones del cliente, se planifica el viaje, y se calcula el importe total del mismo. A continuación, se envía la planificación al cliente y se carga en su tarjeta de crédito el importe del viaje, pasando la información al departamento de contabilidad, que se encargará del pago de los diferentes servicios contratados. Con esto, el proceso iniciado por la agencia de viajes finaliza. En caso de que alguna de las reservas no pueda realizarse, se contacta con el cliente y se le notifica que la planificación elegida no puede llevarse a cabo. Si el cliente desea hacer algún cambio en la misma, se inicia un nuevo proceso de reserva por parte de la agencia. La figura 5 muestra el modelo de FT definido.

5 Generación Automática de Código

A partir de la información sobre el modelo de FT almacenada en el repositorio, es posible obtener versiones del mismo, ejecutables en diferentes entornos. La generación de código se hace a nivel de proyecto, es decir, no se genera automáticamente código para un proceso concreto, sino para el conjunto de procesos que forman un proyecto. El proceso de generación automática se ve interrumpido en caso de que se detecten errores en el FT diseñado. Si no hay errores, se genera un archivo de texto que contiene la especificación del FT. A continuación se enumeran,

de forma resumida, las pautas de traducción seguidas por el generador de código a R-OASIS (una descripción detallada del lenguaje se pueden consultar en [7]). Las palabras en negrita se corresponden con palabras clave del lenguaje y las que aparecen en cursiva se cambiarán por el valor correspondiente. Los distintos patrones de generación de código están directamente relacionados con la formalización del metamodelo de FT.

- Cada proyecto equivale a un esquema conceptual. El literal *Definiciones_de_clases* se sustituye por la generación de código correspondiente a todas las clases del proyecto.

```
conceptual_schema Nombre_Proyecto
domains string; bool; nat.
    Definiciones_de_clases
end_conceptual_schema
```

- Cada proceso del proyecto se corresponde con una clase especializada de la clase proceso genérico. La especificación OASIS de la clase `process` define un clase abstracta que proporciona el patrón genérico de comportamiento y estructura que debe tener cualquier proceso. Para especificar un proceso concreto, utilizamos el operador de especialización y redefinimos en la sección `processes`, las condiciones de inicio y finalización, así como el subproceso `actions`.

  Si observamos el patrón de generación, lo único que cambia de unos procesos a otros son los literales *Nombre_Proceso, start_condition, end_condition* y *Secuencia_actividades_ y_condiciones.* Éste último se sustituye por la secuencia de creación de las actividades y subprocesos *(actividad::start(_Oid,Cod_Act) ó subprocess::start(_Oid,Cod_SubP)),* empezando por la actividad inicial y acabando en la actividad final. La secuencia de invocación se hace de acuerdo al flujo de control y condiciones de transición modeladas en el proceso.

```
complex_class Nombre_Proceso specialization_of process
 processes
  Nombre_Proceso <- start & body & stop.
  body <- check(start_condition) & run & actions & end_run &
         (check(end_condition) & finish or terminate).
  actions <- Secuencia_actividades_y_condiciones.
end_complex_class
```

- Cada actividad del proyecto se corresponde con una clase especializada de la clase actividad. Puesto que existen distintos tipos de actividades, se especializará de la clase actividad correspondiente según el metamodelo. En cualquier caso, el código que se genera sigue el mismo patrón los distintos tipos de actividades. A continuación se muestra la generación de código para una actividad automática.

```
complex_class Nombre_Actividad specialization_of
automatic_activity
processes
   Nombre_Actividad <- start & body & stop.
   body <-  check(start_condition) & run & actions & end_run &
              (check(end_condition) & finish or terminate).
   actions <- resources & actions_activity.
```

```
    resources <- asociación_de_los_recursos_que_utiliza
    actions_activity <- Campo_acciones_de_la_BD.
end_complex_class
```

En este caso, el literal *asociación de los recursos que utiliza* se debe sustituir por la creación y asociación del recurso correspondiente utilizado por dicha actividad. El literal *Campo acciones de la BD* se debe sustituir por las acciones de la actividad, que se definen al crearla o modificarla en la herramienta. El tipo de actividad sólo afecta al tipo de recursos que pueden tener asociados. Al igual que en el caso anterior, *start_condition* y *end_condition* se sustituye por las condiciones correspondientes.

- Todo subproceso es un proceso; por lo tanto, el patrón de generación seguido es el descrito anteriormente para los procesos.
- Cada recurso asociado a algún proceso del proyecto sigue el mismo patrón a la hora de generar código. Al igual que en el caso de procesos y actividades, se genera una clase especializada de la clase recurso correspondiente. Una particularidad al modelar los recursos en la herramienta es poder definir nuevas propiedades asociadas a los mismos. Esto se tiene en cuenta en la generación, de forma que para cada nueva propiedad que se añada al recurso, se añadirá una definición de atributo constante en la clase especializada que se genere. A continuación se muestra el patrón seguido para el recurso aplicación.

```
complex_class Nombre_Apli specialization_of application
  constant_attributes
       Nombre_Nueva_Propiedad1(Tipo_Nueva_Propiedad1);
       ...
       Nombre_Nueva_Propiedad1(Tipo_Nueva_Propiedad1).
end_complex_class
```

- El patrón de generación de código para las asignaciones de recursos a entidades consta de dos partes: a) creación del objeto que representa al recurso, invocando el evento de creación de la clase correspondiente; b) asignación de dicho recurso a la entidad correspondiente, creando una instancia del objeto agregado que representa dicha asociación, según las clases del metamodelo definido. Por ejemplo, para asociar una aplicación a una actividad automática, el código generado es:

```
a)Nombre :: new_app(Oid,Cod_Apli,Nombre,Desc,Path,Params,
                    Perms,[],'')
b)invokesAppAAc :: new_invokesAppAAc(Oid,Cod_Apli,Nombre,
          Desc,Path,Params, Perms,[],'',[Cod_Act,Cod_Apli])
```

De acuerdo con los patrones descritos y siguiendo con el ejemplo de la agencia de viajes, la figura 6 muestra la generación de código obtenida para la clase `reservaViaje` que representa el proceso global y la clase `cargoTarjetaCrédito` que representa una actividad automática del proceso.

```
conceptual_schema PY0016
domains string; bool; nat.
 …
complex_class reservaViaje specialization_of process
processes
 reservaViaje <- start & body & stop.
 body <- check(true) & run & actions & end_run & (check(true) &
         finish or terminate).
 actions<-I::start(_Oid,'AC0149') &
  (reservaHoteles::start(_Oid,'AC0151') &
   reservaTransporte::start(_Oid,'AC0155') &
   reservaCoches::start(_Oid,'AC0153')) &
   (check(ConfH=True & ConfT=true & ConfC=true)&
    planificacionViaje::start(_Oid,'AC0156') &
   (envioplanificación::start(_Oid,'AC0157') &
    (cargoTarjetaCrédito::start(_Oid,'AC0158') &
     envioContabilidad::start(_Oid,'AC0160')))
   )) or
   contactarcliente :: start(_Oid,'AC0159') &
   F :: start(_Oid,'AC0150').
end_complex_class

complex_class cargoTarjetaCrédito specialization_of
automatic_activity
 processes
   cargoTarjetaCrédito <- start & body & stop.
   body <-  check(true) & run & actions & end_run &(check(true) &
            finish or terminate).
   actions <- resources & actions_activity.
   resources <- aliasOid(factura(proceso('PR0014')),OidFactura)&
      usesDataAc::new_usesDataAc(_OID2,'AC0158_OidFactura','','',
                                   ['AC0158','OidFactura']) &
      cargoVISA :: new_app(_OID3,'AP0010','CargoVISA','','','','',[],'')&
     invokesAppAAc::new_invokesAppAAc(_OID4,'AC0158_AP0010','','',
                                  ['AC0158','AP0010']).
   actions_activity <- OID4::invokesapp('CargoVisa',_OID1,_).
 end_complex_class
  ...
 end_conceptual_schema.
```

Figura 6. Código R-OASIS generado

6 Conclusiones y Trabajos Futuros

Se ha presentado un entorno para la definición y validación de modelos de flujo de trabajo. El entorno permite al analista construir un modelo gráfico que representa el proceso seguido por una organización, de acuerdo a un metamodelo de referencia. El modelo construido se puede validar en un entorno de prototipación automática, puesto que hay un soporte formal. Precisamente es la formalización del metamodelo la que

permite tener un generador automático de código que obtiene la especificación formal equivalente al modelo construido. Una vez validado el modelo de flujo de trabajo, se obtiene una versión ejecutable en un entorno de ejecución de flujos de trabajo eficiente. Un generador implementa las equivalencias entre ambos metamodelos. Actualmente se están desarrollando generadores para el sistema OPERA y el sistema MQSeries WF.

Como trabajos futuros se pueden citar algunas cuestiones tanto referentes a posibles mejoras del entorno actual, como a la inclusión de nuevas funcionalidades. Se puede mejorar la ergonomía de la herramienta, dando mayores facilidades al usuario para poder visualizar gráficamente las distintas dimensiones del flujo de trabajo que está modelando, puesto que actualmente hay un única vista que integra todas las dimensiones. Dar opción a que cuando al usuario introduzca nuevas propiedades de los recursos, éstas puedan ser considerados como atributos variables, pudiendo modificar su valor en tiempo de ejecución de acuerdo con el acontecer de ciertos eventos. Finalmente, una funcionalidad interesante sería que el entorno, a través del modelo gráfico que se esté validando, o bien, ejecutando en sistemas reales, pueda visualizar los cambios de estado que se produzcan en los componentes del proceso, mostrándose como avanza la ejecución del mismo.

Bibliografía

[1] Action Workflow. http://www.actiontech.com

[2] G.Alonso, D. Agrawal, A. El Abbadi, C.Mohan. "Funcionality and Limitations of current workflow Management systems", IEEE-Expert, 1997.

[3] Alonso, G., and Mohan, C., Workflow Management Systems: The next generation of distributed processing tools. In "Advanced Transaction Models and Architectures", S. Jajodia and L. Kerschberg (Eds.), Kluwer Academic Publishers, 1997, pp. 35--62.

[4] Borland Delphi 4. Manual de usuario, 1999.

[5] Booch, G., Rumbaugh, J., Jacobson, I., *El lenguaje unificado de modelado*. Addison-Wesley, 1999.

[6] Canós, J. H., Penadés, M.C., Ramos, I., Pastor, O., *A knowledge-base architecture for object societies*, Proc. of the DEXA-95 Workshop, OMNIPRESS, 1995.

[7] Canós, J.H.. *"OASIS como lenguaje único para Bases de Datos Orientadas a Objetos"*, Tesis Doctoral. Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de Valencia, 1996.

[8] Canós, J.H., Sánchez, J, Penadés, M.C., *Una aproximación ascendente al modelado de flujos de trabajo*. Actas V Jornadas de Ingeniería del Software. Valladolid. Noviembre, 2000.

[9] Georgakopoulos, D., Hornick, M. and Sheth, A., *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure.* Distributed and Parallel Databases, 3, 119-153, 1995.

[10] Hagen, C.J. "A generic kernel for reliable process support". Dissertation. ETH Nr. 13114. ETH Zurich, 1999. (http://www.inf.ethz.ch/department/IS/iks/research/opera.html

[11] Hollingsworth, D., The Workflow Reference Model, Technical Report TC00-1003, WfMC, 1994. http://www.wfmc.org/

[12] Leyman,F, Roller,D., *Production Workflow. Concepts and Techniques*. Prentice Hall, 2000

[13] Lotus Notes. http://www.lotus.com

[14] IBM MQSeries Workflow . http://www.redbooks.ibm.com

[15] Pastor, O.,Ramos, I. *OASIS version 2 (2.2): A Class-Definition Language to Model Information Systems using an Object-Oriented Approach*, SPUPV-95.788, Universidad Politécnica de Valencia, 1995.

[16] Penadés, M.C., Canós, J.H., Carsí, J.A. *Hacia una herramienta de soporte al proceso software basada en la tecnología de workflow,* Actas de las IV Jornadas de Ingeniería del Software. Universidad de Extremadura. Noviembre, 1999.

[17] Penadés, M.C., Canós, J.H. *Modelado conceptual de flujos de trabajo,* Technical Report DSIC-II/8/00, Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, 2000.

[18] Penadés, M.C., Canós, J.H., Sánchez, J. *Automatic Derivation of Workflow Specifications from Use Case Models*. Technical Report DSIC-II/4/01 , Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, 2001.

[19] Segura, M.Paz, *Representación gráfica de FT y generación automática de código en R-OASIS.* PFC Facultad de Informática, M.Carmen Penadés, Valencia, Septiembre, 2000. http://www.bdpfc.inf.upv.es

[20] Sheth, A., Georgakopoulos, D., Joosten, S., Rusinkiewicz, M., Scacchi, W., Wileden, J. and Wolf., A. *Report from the NSF Workshop on workflow and Process Automation in Information Systems*. Computer Science Department Technical Report, UGA-CS-TR-96-003, University of Georgia, October 1996. http://lsdis.cs.uga.edu/activities/NSF-workflow/final-report.ps.