

# JISBD2007-03: Biological Data Processing using Model Driven Engineerings

A. Gómez, A. Boronat, J.Á. Carsí, I. Ramos, C. Täubner y S. Eckstein

**Abstract**— This paper shows how Model-Driven Software Development (MDSD) can be applied in the bioinformatics field since biological data structures can be easily expressed by means of models. The existence of several heterogeneous data sources is usual in the bioinformatics context. In order to validate the information stored in these data sources, several formalisms and simulation tools have been adopted. The process of importing data from the source databases and introducing it in the simulation tools is usually done by hand. This work describes how to overcome this drawback by applying MDSD techniques (e.g. model transformations). Such techniques allow us to automate the data migration process between source databases and simulation tools, making the transformation process independent of the data persistence format, obtaining more modular tools and generating traceability information automatically.

**Keywords**— Bioinformatics, Model Driven Software Development -MDSD-, data migration, biological systems modeling.

## I. MOTIVACIÓN.

LA práctica tradicional en los estudios científicos de «experimentación → análisis → publicación» está cambiando a «experimentación → organización de datos → análisis → publicación». Esto se debe a que, hoy en día, los datos no se obtienen únicamente de experimentos, sino que también se obtienen de simulaciones. Esta gran cantidad de datos generados no siempre tiene la misma estructura y puede estar almacenada en bases de datos heterogéneas. Además, el elevado número de datos requiere el desarrollo de herramientas informáticas que nos permitan representarlos para analizarlos, y a su vez, realizar otras simulaciones con ellos.

---

Este artículo ha sido financiado por el Proyecto Nacional de Investigación Científica, Desarrollo e Innovación META TIN2006-15175-C05-01, la ayuda complementaria de la Generalitat Valenciana ref. ACOMP07/216, y a la beca del programa de Formación de Profesorado Universitario, ref. AP-2006-00690.

A. Gómez, J. Á. Carsí, e I. Ramos pertenecen al Departamento de Sistemas Informáticos y Computación en la Universidad Politécnica de Valencia, Cno. de Vera s/n, 46022 Valencia. (correos e.: agomez@dsic.upv.es, pcarsi@dsic.upv.es, iramos@dsic.upv.es).

A. Boronat pertenece en la actualidad al *Department of Computer Science*, F28 Mathematics & Computer Science Building, University of Leicester, University Road, Leicester, LE1 7RH, United Kingdom. (correo e.: aboronat@mcs.le.ac.uk).

C. Täubner y S. Eckstein pertenecen al Institut für Informationssysteme, Technische Universität Braunschweig, Mühlentpfordtstraße 23, 2. OG, D-38106 Braunschweig, Deutschland. (correo e.: c.taebner@tu-bs.de, s.eckstein@tu-bs.de).

En el campo de la bioinformática encontramos esta problemática en el análisis y simulación de los mecanismos de señales de las células (*pathways*). Un *pathway* es un conjunto de reacciones químicas que se producen en el interior de una célula tras la recepción de un estímulo. En estudios de este tipo se observa tanto el rellenado independiente de fuentes de datos como el desarrollo independiente de herramientas de modelado. Por ello, estos datos deben ser reconvertidos de forma manual para que puedan ser utilizados en las herramientas de simulación. En un escenario así, disponer de herramientas interoperables es deseable.

El Desarrollo de Software Dirigido por Modelos — DSDM — es una aproximación que pretende dar solución a problemas como éste. En el DSDM, un modelo es una estructura de datos que puede ser definida mediante un lenguaje de modelado (generalmente llamado metamodelo). Un modelo permite definir la funcionalidad, estructura y/o comportamiento de un sistema [1] dependiendo del metamodelo utilizado. La utilización de modelos en un proceso de DSDM permite automatizar el desarrollo de aplicaciones y su evolución mediante técnicas de programación generativa [2], como las transformaciones de modelos y la generación de código.

Este artículo muestra cómo la filosofía de DSDM permite resolver los problemas surgidos en el estudio de los caminos de señales en el campo de la bioinformática. Los problemas de interoperabilidad entre aplicaciones pueden abordarse de una manera sistemática, donde (a) el formato de datos puede definirse como un modelo y (b) los datos se definen como una colección de objetos que son instancia de las clases de este modelo. El tratamiento de los datos desde la perspectiva del DSDM permite (c) el desarrollo de herramientas donde el tratamiento de éstos sea independiente de los formatos de persistencia, obteniendo herramientas más modulares. Esto a su vez facilita (d) la automatización en la migración de los datos mediante el uso de técnicas de transformaciones de modelos. Todos estos factores reducen el coste de producción de las herramientas afectando de forma directa y positiva en la productividad de los usuarios/biólogos.

El documento se organiza de la siguiente manera: en el apartado II se expone el contexto biológico, introduciendo las razones por las que es interesante el estudio de los caminos de señales (*pathways*), y se describe la aproximación actual, en la que esto es un proceso poco eficiente. El apartado III

introduce las bases del desarrollo dirigido por modelos y la tecnología empleada para llevar a cabo la migración de datos. En el apartado IV se describe la solución diseñada para resolver las deficiencias de la aproximación actual, y por último en el apartado V se exponen las conclusiones finales y trabajos futuros.

## II. CASO DE ESTUDIO

Numerosos procesos biológicos son controlados por los denominados mecanismos de señales. Un camino de señales —*signaling pathway*— es una colección de reacciones químicas que tienen lugar en una célula como reacción a un estímulo<sup>1</sup> (generalmente externo) disparando diversos eventos en el interior de ésta. Estos eventos son los que controlan la progresión del ciclo de vida (muerte, crecimiento, especialización), activación o desactivación de genes para el momento de la reproducción, el movimiento de la célula, etc. El estudio de estos caminos de señales se realiza mediante el ensayo empírico de los estímulos externos que recibe una célula y recogiendo los datos sobre las reacciones químicas desencadenadas en su interior. Estos datos obtenidos de forma experimental por los biólogos son almacenados en diferentes bases de datos para su posterior estudio y consulta por otros expertos. Existen diversas fuentes donde es almacenada la información como por ejemplo TRANSPATH®, BioCarta, KEGG o Reactome.

Las reacciones químicas involucradas en un *pathway* suelen ser un proceso en cascada, produciéndose entonces reacciones de forma concurrente. Por ello, para el modelado y análisis de estos procesos biológicos se han adaptado y aplicado algunos de los formalismos desarrollados para el análisis de sistemas concurrentes. Ejemplo de ello son aproximaciones en  $\pi$ -cálculo [3], ambient calculus [3], *Life Sequence Charts* (LSCs) [5] o redes de Petri [6].

El caso de estudio parte de los trabajos realizados por Taübner et al. [7], donde se ha modelado, simulado y validado el *pathway* del receptor de tipo toll 4 (TLR4) —que se introduce en el siguiente punto— mediante redes de Petri coloreadas. A continuación se expone en primer lugar el contexto biológico describiendo el problema abordado en [7] y en segundo lugar se describen los detalles tecnológicos de éste: fuente de datos, herramienta de simulación seleccionadas y proceso de extracción y tratamiento de los datos.

### A. Los receptores de tipo Toll. El *pathway* TLR4.

Los receptores de tipo Toll —*Toll-like receptors* (TLR)— son una familia de proteínas que forman parte del sistema inmunitario. Permiten la adaptación del sistema inmune de diversos seres vivos siendo responsables del reconocimiento de diversos patrones (secuencias de moléculas) que identifican a diferentes patógenos. Una vez reconocido el patógeno (cualquier entidad biológica capaz de producir daño) estimulan la respuesta de defensa contra él por parte del sistema inmunitario.

Actualmente se han identificado trece receptores de tipo Toll en humanos (TLR1-TLR13). Estos receptores reconocen moléculas que están constantemente asociadas a amenazas contra el sistema inmune y que son muy específicas a estas amenazas, no pudiendo confundirlas con moléculas propias. Algunas de estas moléculas específicas pueden ser los lipopolisacáridos (LPS) presentes en la superficie de células bacterianas, proteínas presentes en los flagelos de bacterias, ARN de doble cadena presente en virus, etc.

Para el caso de estudio, por simplicidad, se toma como ejemplo una pequeña parte del *pathway* en el que interviene el receptor de tipo Toll 4, por ser el primero reconocido en mamíferos y el mejor estudiado hasta el momento. A continuación se indican las reacciones que serán transformadas en el caso de estudio.

1.  $LPS + LBP \rightleftharpoons LPS:LBP$
2.  $LPS:LBP + CD14 \rightleftharpoons LPS:LBP:CD14$
3.  $ST2 + TIRAP \rightleftharpoons ST2:TIRAP$
4.  $ST2 + MyD88 \rightleftharpoons ST2:MyD88$

La notación expresa a la izquierda los reactivos y a la derecha los productos. El símbolo « $\rightleftharpoons$ » expresa que esta reacción es bidireccional, ya que se trata de una reacción de equilibrio químico —informalmente se puede describir como una reacción que se mantiene hasta que los reactivos y los productos alcanzan un estado de equilibrio, no llegándose a consumir ninguno de los dos—. Para (1), por ejemplo, podríamos expresar de forma sencilla lo siguiente: la molécula LPS se une a la molécula LBP, dando lugar a la molécula compuesta LPS:LBP.

### B. Aproximación actual en el estudio del *pathway* del TLR4 en bioinformática.

Son numerosas las fuentes de datos de las que se puede extraer información sobre el *pathway* del TLR4. Algunas de ellas pueden ser TRANSPATH®, KEGG, Reactome o BioCarta. Estas bases de datos han sido rellenadas a partir de estudios empíricos y pueden contener información incompleta e incluso contradictoria. Los actuales trabajos para la representación y simulación de *pathways* se dirigen en la dirección de poder validar la información obtenida en los estudios empíricos.

Con ese objetivo se han adaptado diversos formalismos gráficos para la simulación de *pathways* biológicos. Ejemplos de ello son los diagramas de estados [3], los *Life Sequence Charts* [3][5], o las redes de Petri coloreadas [7]. El trabajo en el que se fundamenta la solución aquí presentada describe un proceso mediante el cual es posible extraer los datos del *pathway* TLR4 de la base de datos TRANSPATH® con el objetivo de simularlo mediante la herramienta de simulación y validación de redes de Petri coloreadas *CPN Tools*. TRANSPATH® es una base de datos que almacena información acerca de caminos de señales. Permite la exportación de datos en XML, y en su versión 7.4 contiene información sobre 62.549 moléculas, 23.076 genes y 104.362 reacciones. Una red de Petri, por su parte, es una representación formal para un sistema distribuido discreto,

<sup>1</sup> Un estímulo puede ser, por ejemplo, la presencia de una determinada molécula en el entorno de la célula.

que permite representar eventos concurrentes. La red se forma por nodos (llamados lugares), transiciones y arcos dirigidos. Estos arcos conectan siempre un lugar con una transición o una transición con un lugar. En los lugares puede haber un determinado número de *tokens*, que pueden moverse de un lugar a otro cuando una transición se dispara (una transición se habilita cuando todas sus entradas contienen *tokens*). La Fig. 1 muestra un ejemplo de red de Petri, donde los círculos blancos son los lugares, el rectángulo relleno de color negro es una transición, las flechas son los arcos dirigidos, y los puntos negros, los *tokens*.

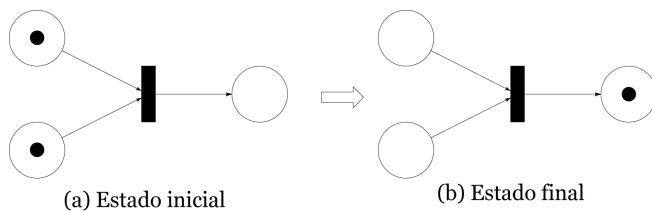


Fig. 1. Ejemplo de red de Petri.

Una red de Petri coloreada es aquella en la que los *tokens* pueden tener un determinado color (esto es, algún atributo distintivo), pudiendo caracterizar *tokens* de diferentes tipos. *CPN Tools* es una herramienta que permite la construcción, modificación y validación sintáctica de redes de Petri coloreadas de forma gráfica. Dispone además de un simulador (tanto interactivo como automático) para poder inspeccionar la evolución de estos modelos.

En [7] el método empleado para poder realizar la simulación a partir de los datos almacenados en TRANSPATH® es manual. Esto supone que el usuario/biólogo que va a realizar la simulación debe consultar de forma manual la base de datos obteniendo el listado de reacciones que intervienen en el *pathway* que desea estudiar. Con los datos obtenidos debe crear y editar de forma manual la correspondiente red de Petri coloreada en *CPN Tools*, creando uno a uno cada uno de los lugares, transiciones, arcos, *tokens*, etc. En el trabajo al que se hace referencia, esto supuso crear de forma manual —y únicamente para el ejemplo del *pathway* del TLR4— 75 lugares, 47 transiciones y casi un centenar de colores.

### III. INGENIERÍA DIRIGIDA POR MODELOS

La Ingeniería Dirigida por Modelos es un campo en la Ingeniería del Software que, durante años, ha representado los artefactos software como modelos con el objetivo de incrementar la productividad, calidad, y reducir los gastos en el proceso de desarrollo de software. Recientemente, existe un interés creciente en este campo. Prueba de ello es la aproximación de *Model Driven Architecture* [1], apoyada por el OMG.

El Desarrollo Dirigido por Modelos ha evolucionado del campo de la Ingeniería Dirigida por Modelos. En él, no sólo las tareas de diseño y generación de código están involucradas, sino que también se incluyen las capacidades de trazabilidad, tareas de meta-modelado, intercambio y

persistencia de modelos, etc. Para poder abordar estas tareas, las operaciones entre modelos, transformaciones, y consultas sobre ellos son problemas relevantes que deben ser resueltos. En el contexto de MDA se abordan desde el punto de vista de los estándares abiertos. En este caso, el estándar *Meta Object Facility* (MOF) [9], proporciona un mecanismo para definir metamodelos. El estándar *Query/Views/Transformations* (QVT) [10] indica cómo proporcionar soporte tanto para transformaciones como para consultas. A diferencia de otros lenguajes nuevos, QVT se apoya en el ya existente lenguaje *Object Constraint Language* (OCL) para realizar las consultas sobre los artefactos software.

#### A. MOMENT. Un framework para la Gestión de Modelos.

MOMENT [11] es una herramienta que da soporte a los estándares propuestos por el OMG para dar soporte a transformaciones. La herramienta proporciona un soporte algebraico para las tareas de transformación y consulta de modelos mediante un eficiente sistema de reescritura de términos —*Maude*— y desde un entorno de modelado industrial —*Eclipse Modeling Framework* (EMF)—. EMF [12] puede ser visto como una implementación del estándar MOF, y permite la importación automática de artefactos software desde orígenes de datos heterogéneos: modelos UML, esquemas XML, etc. Respecto a *Maude*, MOMENT aprovecha las capacidades de modularidad y parametrización de este sistema para proporcionar un entorno de transformación y consulta de modelos de forma genérica e independiente de metamodelo.

Como lenguaje para transformaciones MOMENT se apoya sobre el estándar abierto QVT, proporcionando una implementación tanto del lenguaje QVT-Relations como de OCL. QVT-Relations es un lenguaje de transformaciones declarativo que proporciona de forma implícita capacidades de trazabilidad. Para este lenguaje, MOMENT ofrece un amplio soporte para transformaciones unidireccionales 1-a-1. Además, esta herramienta proporciona una completa implementación de los operadores de consulta del lenguaje OCL.

### IV. UN ENFOQUE DE DSDM EN LA MIGRACIÓN DE DATOS BIOLÓGICOS

En el trabajo inicial sobre el estudio del *pathway* del TLR4, la migración de datos desde la base de datos origen hasta la herramienta de simulación para su representación mediante una red de Petri coloreada debe realizarse de forma manual. A continuación se muestra una solución al problema de migración de datos del caso de estudio empleando transformaciones, según la filosofía de desarrollo de software dirigido por modelos. Esto supone las siguientes tareas: (a) desarrollo del modelo de datos del dominio origen (TRANSPATH®) y (b) desarrollo del modelo de datos del dominio destino (CPN Tools). (c) Definición mediante un lenguaje de transformaciones las reglas de transformación entre el dominio origen y el dominio destino; (d) implementación del mecanismo de preprocesado de datos

permitiendo reconstruir los datos originales como instancias del modelo origen, y (e) definición del postprocesado de los datos, que implementa el mecanismo de persistencia al formato final.

A continuación se presenta la solución proporcionada. En primer lugar, se describe el proceso de transformación y sus etapas, en segundo lugar se describen los modelos del dominio origen y el modelo destino, y por último, se describe en mayor detalle el proceso de transformación.

#### A.Arquitectura e implementación de la herramienta.

El proceso de migración de datos, tal y como se deriva de las tareas anteriores, se realiza en tres pasos: (1) recuperación y preprocesado de los datos, (2) ejecución de la transformación mediante un motor de transformaciones y (3) postprocesado y persistencia de los datos. En una aproximación de DSDM, el uso de un motor de transformaciones implica que se deben definir en primer lugar los modelos origen y destino de la transformación para poder establecer las correspondencias entre uno y otro posteriormente.

La solución aquí presentada hace uso de MOMENT. MOMENT es una herramienta integrada en el entorno *Eclipse* que proporciona soporte para transformaciones. Como entorno de modelado, MOMENT emplea el *Eclipse Modeling Framework* (EMF). Este framework proporciona como lenguaje de modelado *Ecore*. Además, emplea como formato de persistencia XMI. EMF permite —entre otros— la creación de modelos *Ecore* a partir de un esquema XSD. No obstante, los modelos *Ecore* obtenidos a partir de esquemas XSD son complejos y no siempre representan la semántica de los datos de forma clara. Por ello, se ha decidido definir de forma manual los modelos origen y destino, teniendo sólo en cuenta aquella información relevante para el caso de estudio.

La Fig. 2 muestra la arquitectura de la herramienta. En ella se reflejan de forma clara los 3 pasos necesarios para realizar la migración de los datos. En primer lugar se extraen los datos de la base de datos TRANSPATH® (A), y se reconstruyen como una instancia en XMI (B) del modelo *Ecore* de TRANSPATH® (C). Este primer paso (1) se realiza de forma sencilla en Java ya que la correspondencia de los datos origen con el modelo en EMF es directa. La implementación de este primer paso de preprocesado de los datos es una adaptación del trabajo realizado en [13].

El segundo paso de la migración (2) es el principal y más complejo. Se realiza mediante MOMENT y su motor de transformaciones, ejecutando la transformación del dominio de TRANSPATH® (reacciones, moléculas, etc.) al dominio de CPN Tools (lugares, transiciones, arcos, etc.). Tras la definición de las reglas de transformación (D) entre los dominios origen (C) y destino (E), se aplica la transformación sobre los datos recuperados de la base de datos (B) obteniendo una instancia en el dominio de CPN Tools (F). Por último, el tercer paso de la migración (3) es de nuevo un proceso trivial en el que se persisten los datos desde EMF (F) a un fichero XML comprensible por la herramienta *CPN Tools* (G). En

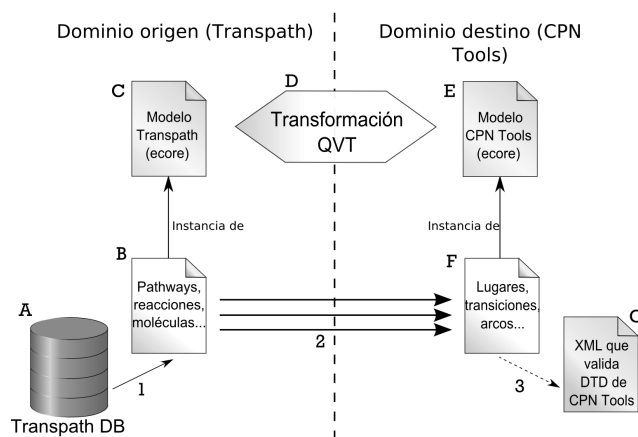


Fig. 2. Arquitectura de la herramienta.

este, paso además, pueden realizarse otras tareas de postprocesado, como por ejemplo, la inclusión de algún algoritmo de redibujado (*layout*) sobre los elementos de la red de Petri.

#### B.Desarrollo de los modelos.

En primer lugar se ha definido un modelo únicamente con las partes que resultan de interés para la simulación de un *pathway* eliminando los conceptos innecesarios de la compleja base de datos TRANSPATH®. La Fig. 3 muestra —mediante una metáfora visual similar al diagrama de clases de UML— el modelo *Ecore* desarrollado. En el modelo encontramos la clase *Network*, elemento raíz del modelo. Una *Red* —*Network*— contiene un conjunto de *Pathways*, *Reacciones* y *Moléculas*. A su vez, un *Pathway* se compone de diversas Cadenas —*Chains*— de reacciones, y una reacción puede estar involucrada en diferentes cadenas. Por último, las reacciones se relacionan con las moléculas. Una molécula puede ser un reactante de una reacción, puede ser un producto, o puede intervenir indirectamente como catalizador o inhibidor. Las clases *ReactantsCoefficient*, *ProductsCoefficient*, *EnzymesCo-efficient* e *InhibitorsCoefficient* heredan todas de la clase *Coefficient*, omitida por motivos de claridad, que contiene un atributo *coefficient* de tipo entero. Estas clases permiten expresar que una molécula interviene con un cierto coeficiente<sup>2</sup> en una reacción, e intentan representar, según la expresividad de *Ecore*, la semántica de una clase asociación.

La Fig. 4 muestra el modelo creado para la herramienta *CPN Tools*. En este caso, se ha decidido un diseño más alejado del diseño conceptual de una red de Petri Coloreada, incluyendo conceptos específicos de la propia herramienta *CPN Tools*. Esto se debe a que un diseño así que permite tratar con todos los conceptos interesantes de la herramienta *CPN Tools* desde EMF (color de los elementos gráficos y arcos, posiciones, etc.). Además, de esta manera el proceso de persistencia al fichero XML final es trivial, siendo una correspondencia 1-a-1.

<sup>2</sup> El coeficiente representa el número de moléculas que aparecen en la ecuación de una reacción. Por ejemplo, en la reacción  $2\text{H}_2 + \text{O}_2 = 2\text{H}_2\text{O}$ , los coeficientes son los números que aparecen a la izquierda de las moléculas  $\text{H}_2$  y  $\text{H}_2\text{O}$ .



Fig. 3. Modelo de la base de datos Transpath.

El modelo —donde la clase raíz es *Cpnet*— se compone de dos grande bloques de clases: aquellas que cuelgan de *Globber* y aquellas que lo hacen de *Page*.

La Fig. 4 muestra estos dos grande grupos separados mediante una línea discontinua. El primer grupo de clases permite representar las declaraciones de la red (*colores* —*Color sets*—, *variables*, *bloques*, etc.). La herramienta permite definir distintos tipos de colores (*Color Sets*). De todos los tipos que proporciona, por simplicidad (ya que son los únicos necesarios) sólo se han incluido el *Color Set* simple «*Enumerated*» y el *Color Set* compuesto «*Product*».

El segundo grupo de clases (aquellas contenidas en el

elemento *Page*) representan a todos los elementos visuales de la red de Petri coloreada. Estos elementos visuales heredan todos de la clase *DiagramElement*, y además, pueden estar contenidos en distintos grupos —*Group*—. Así, dentro de una página podemos encontrar lugares —*Places*—, transiciones —*Trans*—, arcos —*Arcs*—, anotaciones —*Annot*—, etc. Por su parte, se dice que los lugares que se han definido son de un tipo (color) que debe haber sido definido en las declaraciones (mediante el rol *type* desde la clase *Place* a la clase *ColorSet*). Las clases *InitMark* y *Mark* permiten representar el estado de un determinado lugar, indicando los *tokens* que se encuentran en éste. El tipo de estos *tokens* viene dado por el rol

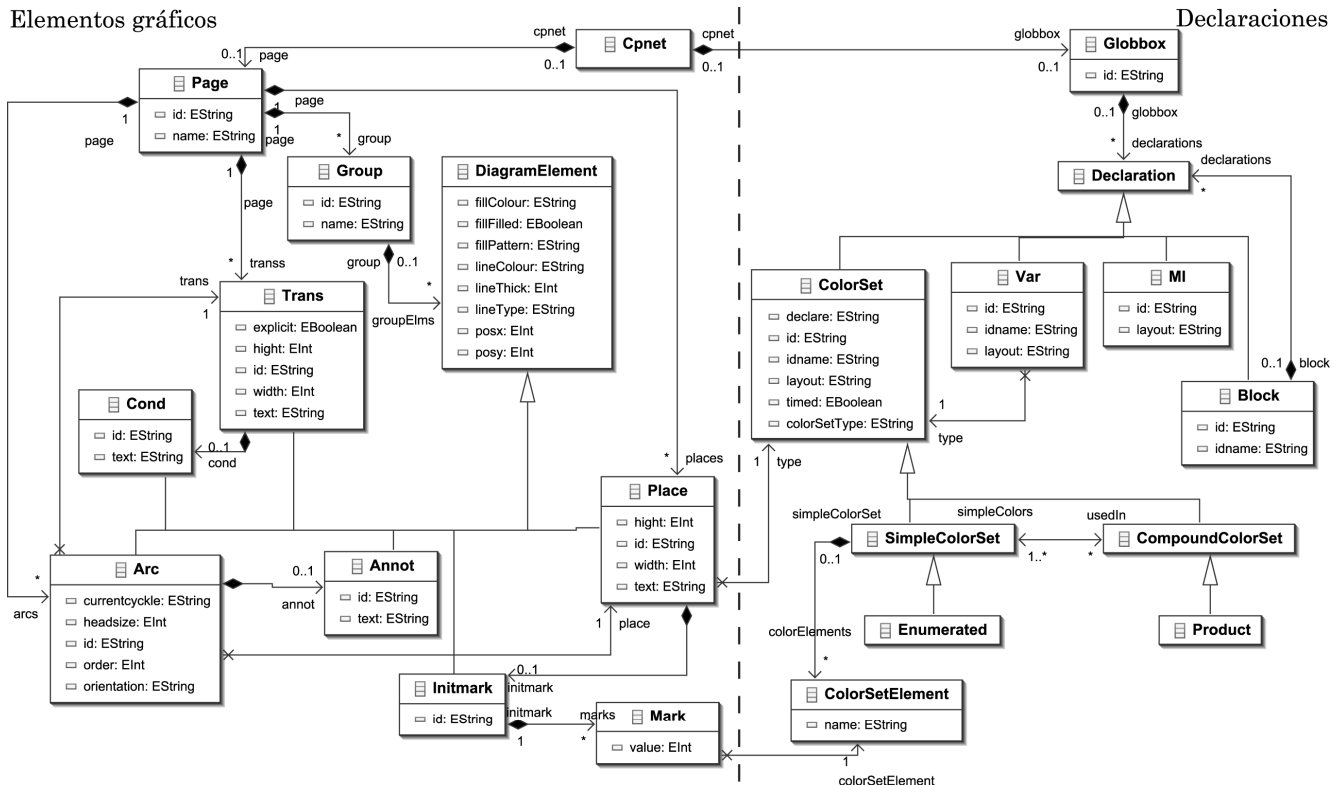


Fig. 4. Modelo de la herramienta CPNTTools.

*colorSetElement* entre las clases *Mark* y *ColorSetElement*.

C.Proceso de transformación.

Finalmente, se han definido las reglas de transformación que permiten convertir los datos del dominio origen al dominio destino. Estas reglas, expresan las correspondencias establecidas por los biólogos —cuando construyen una red de Petri coloreada de forma manual— entre los datos de TRANSPATH® y las primitivas de la herramienta CPN Tools. La Tabla I muestra de forma simplificada estas correspondencias entre el dominio origen y el dominio destino.

TABLA I  
CORRESPONDENCIAS ENTRE EL DOMINIO ORIGEN Y EL DOMINIO DESTINO.

Transpath	CPN Tools
Network	Cpnet
Pathway	Globbox Page
Molecule (complex)	Product
Molecule (simple)	Enumerated
Reaction	Trans
Molecule (reactant)	Place Arc (de Place a Trans)
Molecule (product)	Place Arc (de Trans a Place)

El lenguaje con el que se expresan las relaciones entre ambos dominios es QVT-Relations, que se describe de forma breve a continuación. En QVT-Relations, una transformación son un conjunto de relaciones establecidas entre los dominios participantes en la transformación que deben cumplirse para que ésta sea satisfactoria [10]. Un dominio es una variable tipada que puede corresponderse con algún elemento del modelo que va a transformarse. Éste puede tener un patrón, que puede considerarse como un conjunto de restricciones que deben cumplir los elementos del modelo candidato —modelo sobre el que se aplica la transformación— para que se trate de una correspondencia válida.

Los dominios además, pueden caracterizarse mediante el uso de las palabras clave *checkonly* y *enforce*. Para un dominio *checkonly*, la transformación comprobará que existe una correspondencia válida —que satisfaga el patrón del dominio— en el modelo candidato. En caso de que el dominio destino sea *enforce*, si al ejecutar la transformación no existe ninguna correspondencia posible, se creará un elemento que cumpla con el patrón del dominio.

La relación *ReactantsToPlaces*, muestra un ejemplo de la sintaxis para la definición de relaciones y dominios. Esta relación establece la correspondencia entre una molécula y un lugar. Encontramos dos dominios: *tpDomain* se corresponde con el dominio origen (TRANSPATH®) y *cpnDomain* con el dominio destino (CPN Tools). Para el dominio *tpDomain* se comprueba que exista una molécula con un atributo, *name*, de tipo *String*. Para cada molécula que cumpla este patrón deberá existir (y si no existe, se creará) un *Place* en el dominio destino (*cpnDomain*), cuyo atributo *id* contenga el nombre de

dicha molécula.

Regla 1: Regla ReactantsToPlaces.

```
relation ReactantsToPlaces {
  molecName1 : String;
  checkonly domain tpDomain molecule1 : Molecule
    { name = molecName1 };

  enforce domain cpnDomain place1 : Place
    { id = molecName1 };

  //...

  when
    { IsSimpleMolecule(molecule1); }
  where
    { ReactantsToArcs(molecule1,place1,...); }
}
```

A la aplicación de una regla pueden añadirse pre- y post-condiciones mediante el uso de las cláusulas *when* y *where*. Por ejemplo, la regla *ReactantsToPlaces* sólo se aplicará en caso de que la molécula sea de tipo simple —*IsSimpleMolecule(...)* es una función que mediante una expresión OCL comprueba si la molécula es simple o compuesta—. La cláusula *where* establece que tras la aplicación de la *relation* se procederá a la ejecución de la regla *ReactantsToArcs*.

El proceso de transformación completo se realiza de arriba a abajo, navegando el modelo origen a través de las relaciones de contención, definidas como asociaciones de composición. Esto es, se parte del elemento raíz del modelo origen (*Network*), y se navega hacia abajo (*Network* → *Pathways* → *Chains* → *Reactions* → *Coefficients* → *Molecules*) creando en el modelo destino los elementos correspondientes, tal y como expresa de forma resumida la Tabla I.

Para las declaraciones, la transformación creará un *ColorSet* de tipo *Enumerated* a partir de cada molécula simple. A partir de las moléculas complejas se crearán los *ColorSet* de tipo *Product*, y que estarán formados por los *Enumerated ColorSets* correspondientes a las moléculas simples que los forman.

Para los elementos visuales se parte de un objeto de tipo *Reaction*. Para cada reacción, se crea un objeto de tipo *Trans*. Navegando el rol *reactantsCoefficient* de la clase *Reaction* se obtienen los reactantes. Para cada uno de ellos, se crea un objeto *Place*. Por último ya es posible enlazar cada *place* recién creado con el elemento *Trans* correspondiente mediante un arco —*Arc*— de tipo *PtoT* (*Place to trans*, según la terminología de CPN Tools). Para los productos de la reacción se procede de forma similar, en este caso navegando el rol *productsCoefficient*. Como resultado del proceso de transformación podemos observar la Fig. 5, que muestra la representación de las reacciones del caso de estudio en CPN Tools. En la figura se pueden observar cuatro triángulos numerados. Cada triángulo encierra los elementos generados para la reacción identificada con el mismo número. De esta manera, para la reacción (1) —LPS + LBP ⇌ LPS:LBP—, se

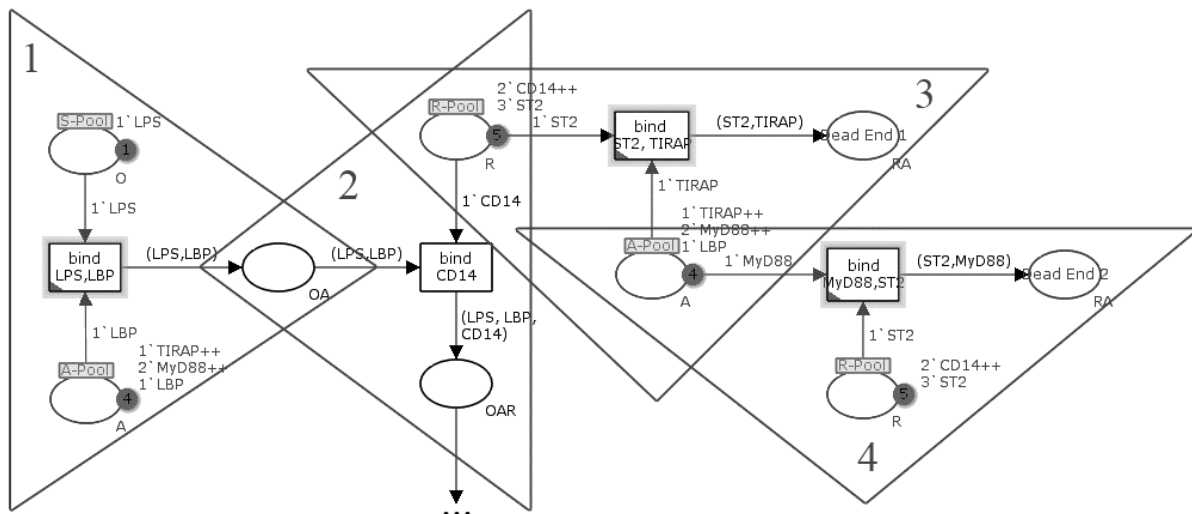


Fig. 5. Representación parcial del Pathway del TLR4 en CPN Tools.

han generado los elementos dentro del triángulo izquierdo (1). De igual manera ocurre con los triángulos 2, 3 y 4, que se corresponden con las consiguientes reacciones.

El código completo de la transformación y los ficheros de ejemplo se encuentran en [14].

## V. CONCLUSIONES

En este artículo se ha presentado un caso de estudio en el cual se aborda un problema de interoperabilidad entre aplicaciones en el campo de la bioinformática mediante un enfoque dirigido por modelos. En este entorno es común la existencia de fuentes de datos y herramientas de simulación heterogéneas, y la natural representación de los datos biológicos como modelos permite abordar estos problemas desde un punto de vista más eficiente (acortando el ciclo de vida del desarrollo de software) y más elegante (puesto que las operaciones se realizan con un lenguaje más expresivo dada su naturaleza declarativa).

El trabajo presenta como ventajas frente a las aproximaciones tradicionales (1) la automatización de tareas que anteriormente se realizaban de forma manual; (2) el desarrollo de herramientas modulares, independizando el mecanismo de transformación del formato de persistencia de los datos y facilitando la extensibilidad y mantenibilidad. (3) Aprovecha las ventajas de las transformaciones de modelos. El uso de modelos para representar los datos a ser transformados permite representar de forma más clara la estructura de éstos, siendo más intuitiva su manipulación ya que se trabaja con conceptos de alto nivel. (4) Se proporcionan capacidades de trazabilidad de forma implícita, ayudando a la localización de información errónea en los orígenes de datos. Por último, (5) el uso de un lenguaje como QVT-Relations ofrece como ventaja (frente a las aproximaciones imperativas tradicionales) que permite expresar de forma declarativa —y por tanto más descriptiva— las correspondencias entre los dominios origen y destino.

Futuras líneas de investigación de esta aproximación dirigida por modelos para el desarrollo de aplicaciones en

bioinformática van dirigidas en dos direcciones. En primer lugar, la representación de los datos biológicos como modelos permite el aprovechamiento de los nuevos *frameworks* para la generación de metáforas visuales a partir de éstos, como es el caso de GMF [9] o MS DSL Tools. Por otra parte las investigaciones en ingeniería dirigida por modelos pueden proporcionar un rico background tecnológico no únicamente para la transformación de datos de un dominio a otro, sino para tareas como la integración de estos obtenidos desde orígenes heterogéneos.

## AGRADECIMIENTOS

Agradecemos al Prof. Dr. H.D. Ehrich y M. Ziegler del *Institut für Informationssysteme* de la *Technische Universität Carolo-Wilhelmina zu Braunschweig*, su apoyo, colaboración y experiencia aportados en el desarrollo de este trabajo. Igualmente agradecemos su duro trabajo a Pascual Queralt como soporte en el uso de MOMENT-QVT.

## REFERENCIAS

- [1] Object Management Group. MDA Guide Version 1.0.1. 2003. Disponible: <http://www.omg.org/docs/omg/03-06-01.pdf>.
- [2] K. Czarnecki and U. W. Eisenecker. Generative programming: methods, tools, and applications. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.
- [3] A. Regev, W. Silverman, and E. Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In Pacific Symposium on Biocomputing, pages 459–470, 2001.
- [4] Regev, Panina, Silverman, Cardelli, and Shapiro. Bioambients: An abstraction for biological compartments. TCS: Theoretical Computer Science, 325, 2004.
- [5] W.Damm and D.Harel. LSCs: Breathing life into message sequence charts. Formal Methods in System Design, 19(1):45–80, 2001.
- [6] M. Heiner, I. Koch, and K. Voss. Analysis and simulation of steady states in metabolic pathways with petri nets, Aug. 02 2001.
- [7] C. Taubner, B. Mathiak, A. Kupfer, N. Fleischer, and S. Eckstein. Modelling and simulation of the TLR4 pathway with coloured petri nets. 2006. EMBS '06. 28th Annual International Conference of the IEEE, pages 2009–2012, August 2006.
- [8] C. Taubner and T. Merker. Discrete modelling of the ethylene-pathway. In ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops, page 1152, Washington, DC, USA, 2005. IEEE Computer Society.

- [9] Object Management Group. Meta Object Facility (MOF) 2.0 Core Specification (ptc/06-01-01), 2006. Disponible: <http://www.omg.org/cgi-bin/doc?formal/2006-01-01>.
- [10] Object Management Group. MOF 2.0 QVT final adopted specification (ptc/05-11-01). 2005. Disponible: <http://www.omg.org/cgi-bin/doc?ptc/2005-11-01>.
- [11] A. Boronat, J. Iborra, J. Ángel Carsí, I. Ramos, y A. Gómez. Del método formal a la aplicación industrial en gestión de modelos: Maude aplicado a eclipse modeling framework. Revista IEEE América Latina, September 2005.
- [12] EMF. <http://download.eclipse.org/tools/emf/scripts/home.php>.
- [13] M. Ziegler. Implementierung eines Transformationsmoduls in Java zur Abbildung von Signaltransduktions-Instanzen auf CPN-Instanzen. Master's thesis, Technische Universität Braunschweig, Braunschweig, Januar 2007.
- [14] A. Gómez. Ficheros de ejemplo para el caso de estudio, 2007. <ftp://moment.dsic.upv.es/transpath2cpn/>
- [15] Eclipse Organization. The graphical modeling framework, 2006. <http://www.eclipse.org/gmf/>.



**Abel Gómez Llana** es Ingeniero en Informática por la Universidad Politécnica de Valencia en 2005. Ha obtenido el título oficial de Máster Universitario en Ingeniería del Software, Métodos Formales, y Sistemas de Información en 2008. En 2005 realizó una estancia en el Instituto de Sistemas de Información (*Institut für Informationssysteme, Technische Universität Braunschweig*) donde inició sus trabajos de investigación sobre bioinformática. Actualmente es Becario FPU por el Ministerio de

Educación y Ciencia en el Departamento de Sistemas Informáticos y Computación de la Universidad politécnica de Valencia. Sus áreas de interés son el desarrollo de software dirigido por modelos, las transformaciones de modelos, y su aplicabilidad en diversos campos como la bioinformática.



**Artur Boronat Moll** es *lecturer* en la Universidad de Leicester, en Reino Unido. Obtuvo el título de Ingeniero en Informática en 2002 y el de Doctor en Informática en 2007, ambos por la Universidad Politécnica de Valencia. Ha realizado estancias en la Universidad de Illinois en Urbana-Champaign (Estados Unidos), y en la actualidad colabora en proyectos de investigación en España, Reino Unido y Estados Unidos en el área de desarrollo de software dirigido por modelos. Sus

áreas de interés son las transformaciones de modelos con reescritura de grafos y la verificación formal de técnicas de desarrollo de software dirigido por modelos.



**José Á. Carsí Cubel** es Licenciado en Informática en 1993 y Doctor en Informática en 1999 por la Universidad Politécnica de Valencia. En la actualidad es profesor Titular de Universidad en la Universidad Politécnica de Valencia siendo miembro del grupo de investigación de Ingeniería del Software y Sistemas de Información del Departamento de Sistemas Informáticos y Computación. Sus actuales reas de interés son: evolución del software, gestión de modelos, lenguajes de descripción de arquitecturas y modelado orientado a objetos.



**Isidro Ramos Salavert** es Catedrático de Universidad en el Área de Lenguajes y Sistemas Informáticos Universidad Politécnica de Valencia (España). 39 años de docencia en distintas universidades españolas y extranjeras.

Ha sido rector fundador de la Universidad de Castilla La Mancha. 20 años como Director de Departamento, de Centro. Director de 23 tesis doctorales. Investigador (principal) de 27 Proyectos de Investigación competitiva (Financiación nacional e Internacional) 3 de ellos activos. Miembro, Presidente Ejecutivo, Presidente de Comité de Programa de numerosas Conferencias y Congresos de investigación en Informática nacionales y extranjeros. Presidente de la Sociedad Española de Ingeniería del Software.

Ha trabajado en: CERN (Suiza), IMAG (Grenoble), CRIN (Nancy), Stony Brook (NYU USA). Coordinador Grupo Investigación PLIS de la UPV. Coordinador Internacional VII Subprograma Electrónica e Informática Aplicadas CYTED Coordinador Proyecto CYTED RITOS. Investigador Proyectos Europeos: BRA ASPIRE (Investigador Principal nodo Valencia).



**Claudia Täubner** se graduó en sistemas de información para empresas en 1998 y obtuvo el grado de doctora en Informática en la Universidad Técnica de Braunschweig (*Technische Universität Braunschweig*) en 2007.

Desde el año 2002 hasta 2007 ha estado trabajando en modelado y simulación de *pathways* de transducción de señales en el grupo de Bioinformática en el Instituto de Sistemas de Información de la Universidad Técnica de Braunschweig. Actualmente trabaja en Volkswagen IT en Wolfsburg.



**Silke Eckstein** obtuvo su grado de doctora en Informática en la Universidad Técnica de Braunschweig (*Technische Universität Braunschweig*) en 2001.

Desde el año 2002 ha trabajado en la consolidación y liderazgo del grupo de Bioinformática en el Instituto de Sistemas de Información en la Universidad Técnica de Braunschweig, instituto del que fue provisionalmente directora desde abril de 2007 a abril de 2008.