Biological Data Migration Using a Model-Driven Approach

Abel Gómez[1.1], José Á. Carsí[1.2], Artur Boronat[1.3], Isidro Ramos[1.4], Claudia Täubner[2.1], and Silke Eckstein[2.2]

[1] Departament de Sistemes Informàtics i Computació, Universitat Politècnica de València. Camino de Vera, s/n. 46022 València. España.
[1.1] agomez@dsic.upv.es, [1.2] pcarsi@dsic.upv.es
[1.3] aboronat@dsic.upv.es [1.4] iramos@dsic.upv.es
[2] Institut für Informationssysteme, Technische Universität Braunschweig, Mühlenpfordtstraße 23, 2.OG, D-38106 Braunschweig. Deutschland.
[2.1] c.taeubner@tu-bs.de, [2.2] s.ecksteinr@tu-bs.de

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. Model-Driven Engineering (MDE), Model-Driven Software Development (MDSD), model transformations, bioinformatics, data migration.

1 Introduction

The traditional sequence of "experiment → analysis → publication" is changing to "experiment → data organization → analysis → publication" [10]. This is because, nowadays, data is not only obtained from experiments, but also from simulations. The great amount of new data that can be generated from these experiments is not always homogeneous and may be stored in different databases. Moreover, the quantity of data requires the development of new computer tools that allow us to represent, analyze, and make new simulations with them.

These problems are also found in the bioinformatics field, especially when analyzing and simulating cell-signaling mechanisms (*Signal Transduction Pathways*). A *signal transduction pathway* is a set of chemical reactions that occur

inside the cell when it receives a stimulus. In studies of this type, it is very common to find both independent databases and modeling tools. Thus, the data of the databases must be converted manually from the source databases to the simulation tools in order to be used. For this scenario, it is desirable to make interoperable tools available.

It would certainly be beneficial to develop different models for a signal transduction pathway using different specification languages to be able to apply different simulation tools. However, this is only possible if the models do not have to be developed by hand but can be generated automatically from the source databases.

Model-Driven Software Development (MDSD) is an approach that attempts to solve problems of this kind. In MDSD, a model is a data structure that can be defined by means of a modeling language (usually called *metamodel*). A model defines the functionality, structure or behaviour of systems [24] depending on the metamodel used. Using models in a MDSD process allows the automation of the development and evolution of the software applications thanks to generative programming techniques [8] such as model transformations and code generation.

This paper shows how the MDSD philosophy can solve the problems that arise in the study of signal transduction pathways in the bioinformatics field. Problems like interoperability between applications can be addressed in a systematic way, where the data structure can be defined by using models and data is defined as a set of objects that are instances of the classes of these models. Dealing with data from the MDSD perspective helps to develop tools where the data processing mechanisms are independent of the final persistence format, obtaining more modular tools. This also helps to automate the data migration process by means of model transformation techniques. All these factors reduce the costs of the software development process, directly increasing the productivity of the users/biologists.

This paper is organized as follows: section 2 explains the biological context, introducing the reason for studying the signal transduction pathways and describes the current approach, which is a very inefficient process. Section 3 introduces the basis of the MDSD and the technologies that have been used to perform the data migration. Section 4 describes the solution proposed to cover the shortcomings of the current approach. Finally, section 5 presents conclusions and future work.

2 Case study

In organisms, proteins have a wide variety of functions and they interact with each other in similar multifaceted ways. These interactions of proteins are described by means of signal transduction pathways or networks, which are typically represented as certain kinds of maps. A distinction is drawn between metabolic and regulatory pathways. Metabolic pathways describe the conversion of classes of substances into other classes of substances, whereas regulatory pathways describe how the function of something is regulated. In this case study,

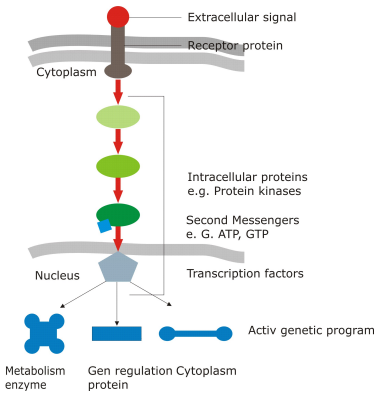

Fig. 1. Signal Transduction (cf. [4])

the conversion of classes of substances into other classes is not significant, but the transduction of signals is (cf. Fig. 1). That is why they are also called signal transduction pathways.

A signal transduction pathway describes how a cell responds to an extracellular signal, e.g. a signaling molecule excreted by a bacterium. The signaling molecule is received at a receptor protein and then transferred via biochemical reactions into the nucleus, where it changes the behaviour that is currently active.

Signal transduction pathways comprise different kinds of molecules: proteins and enzymes with different kinds of functions interact with the help of cofactors, second messengers, phosphatases and small effectors to transmit the signal. The mechanism of transmitting the signal is mediated through state changes of molecules like conformity changes and the building of molecule complexes on the basis of biochemical reactions. These molecule interactions cause the signal flow through the cell and the amplification of the signal in order to reach the nucleus.

Figure 2 shows an example of a signal transduction pathway, where the gray area represents the inside of a cell and the light-colored area represents the outside. The nucleus is represented as a gray ellipse. In this map, molecules are represented with different shapes and colors, which encode the role that a certain molecule plays in the signal transduction pathway under consideration. Examples for such roles are extracellular signals, which are represented as stars; receptors, which are represented as rectangles across the cell membrane; and adapter proteins, which are represented as blue ellipses. Interactions of the molecules appear as lines and arrows, whereas their different shapes stand for different kinds of interactions, e.g. direct or indirect activation or inhibition. Molecules also interact by building molecule complexes, which are represented through narrow cumulations of molecules.

These signal transduction pathways are composed by experts, who study the relevant literature that is produced by various groups worldwide doing research

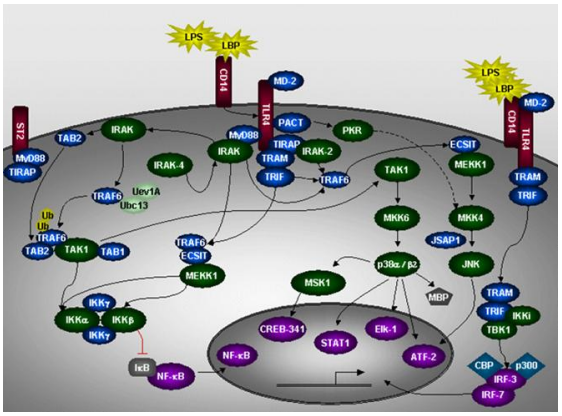

Fig. 2. TLR4 signal transduction pathway in the TRANSPATH® database.

on very small parts of signal transduction pathways in different kind of organisms, e.g. research about short sequences of chemical reactions. This information is then composed bottom-up to a signal transduction pathway and introduced into databases to provide an integrated view on the entire pathway.

Examples for such signal transduction pathway databases are TRANSPATH® [21], KEGG [18], Reactome [17] and BioCyc [19]. They usually provide a web interface for interactive searches and also make their data available as text files in flat file or XML format. Some of the databases already use a more or less standardized exchange format on XML basis, e.g. SBML (Systems Biology Markup Language, [13]).

2.1 Toll-like receptors and the TLR4 signal transduction pathway

In order to give the reader a better understanding of what signal transduction pathways are about, we take the TLR4 signal transduction pathway as an example: Sepsis is the systemic immune response to severe bacterial infection [23]. We are born with a functional, innate immune system that recognizes bacterial and viral products. In sepsis, when a bacterium attacks an endothelial cell, different kinds of mechanisms are activated. Receptors of the innate immune system are activated by microbial components such as LPS (endotoxin, lipopolysaccaride), which is a signaling molecule involved in the initiation of the sepsis syndrome. Receptors, which recognize such LPS molecules, are a family of transmembrane receptors known as Toll-like receptors (TLRs). To date, there are 12 TLRs identified in mice and 10 TLRs identified in humans. TLR4 is one of these and is identified as a significant receptor in mice strain experiments. TLR4 is also annotated in TRANSPATH® and is our example for the explanation of signal transduction pathway (see Fig. 2).

In order to exemplify the transformation of signal transduction pathway data to Petri nets, we must first take a closer look at the biochemical reactions below

that occur at the beginning of the signal transduction pathway: a signal molecule LPS arrives at the cell membrane and binds to the adaptor protein LBP (1) and is delivered to the receptor CD14 (2). This is the beginning of the signaling by TLR4 as mentioned above. The recruitment of the adaptor molecules MyD88 and TIRAP by the TLR4 receptor complex can be inhibited by the sequestering of these critical adaptors during LPS signaling by ST2 (3 and 4, respectively).

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

2.2 An approach to the study of the TLR4 signal transduction pathway

Understanding the flow of information inside a cell is fundamental for an in-depth understanding of the functioning of a cell as a whole. Therefore, modeling and simulating this information flow is beneficial because it helps to understand the flow of signals in a complex network, to test hypotheses in silico before validating them with experiments, and to validate the data collected about a certain signal transduction pathway.

The fact that a flow of information in a complex network must be described has led to the idea of applying languages for the description of concurrent reactive systems in this area, even if these were originally developed to assist the construction or engineering of systems and not the description of already existing systems [11]. A couple of specification languages, such as Petri Nets, Life Sequence Charts, etc., qualify for this task. All of them have different advantages and drawbacks. In the same way, the corresponding simulation tools have different strengths and weaknesses.

We are currently working on one of the major signal transduction pathways databases, TRANSPATH® [21], and we are using Colored Petri Nets [15] among others (e.g. Life Sequence Charts [9], UML-Statecharts, etc.) as the specification language. The corresponding simulation tool is CPN Tools [16]. TRANSPATH® is a database that is accessible by means of the usual methods, i.e., web interface, text files, XML (using its own XML format), etc. In January 2007, TRANSPATH® contained entries about 60,000 molecules, 100,000 chemical reactions, 20,000 genes and 57 signal transduction pathways. The information was based on 30,000 publications. The web interface provides access to all these entries and also contains interactive maps, which give an overview of a certain signal transduction pathway (cf. Fig. 2). The XML version of the database is divided into six files containing data about molecules, genes, reactions, pathways, annotations and references, respectively. They are accompanied by a DTD describing the structure of the files.

Coloured Petri nets are a formal representation for distributed discrete systems that allow concurrent events to be represented. A Petri net consists of two types of nodes (*places* and *transitions*, respectively) and directed arcs. Arcs are

always placed between transitions and places (or places and transitions). Places may contain any number of *tokens*. These tokens can be moved from one *place* to another when a transition is fired (the transitions are enabled if there are tokens in all their input places). Figure 3 shows an example of a Petri net. White circles represent the *places*, black rectangles represent the *transitions*, arrows represent the arcs, and large black dots represent the *tokens*. Coloured Petri nets are an enhancement of Petri nets and can contain different kinds of tokens identified by colors. Now it is possible to represent different dynamic behaviors modeled by different token colors in the same model. *CPN Tools* is a tool for constructing and analyzing coloured Petri nets.

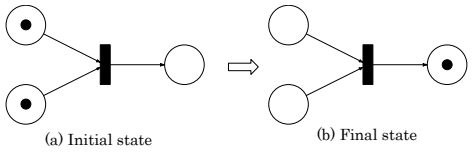

Fig. 3. Petri net example.

In [28], data is extracted from the TRANSPATH® database and introduced in the *CPN Tools* application manually. This implies that the user/biologist who is going to perform the simulation must manually query the database to extract the list of reactions involved in the signal transduction *pathway* to be studied. With the extracted data the corresponding Petri net must be built in the simulation tool manually (creating each one of the places, transitions, arcs, tokens, etc. individually). In [28], this means to manually defining approximately 75 places, 50 transitions, and 100 colours.

3 Model-Driven Engineering

Model-Driven Engineering (MDE) is a Software Engineering field that over the years has represented software artifacts as models in order to increase productivity, quality and to reduce costs in the software development process. Nowadays, there is increasing interest in this field, es demonstrated by the OMG guidelines that support this trend with the Model-Driven Architecture (MDA [24]) approach.

Model-Driven Software Development (MDSD) has evolved from Model-Driven Engineering. MDSD not only involves design and code generation tasks, but also traceability capabilities, meta-modeling features, model persistence and model interchange tasks, etc. To address these tasks, operations between models, transformations, and queries over these models are relevant problems that must be resolved. In the MDA context, they are resolved from the open standards point of view. The Meta Object Facility (MOF) standard [26] provides support for the meta-modeling capabilities. The Query/Views/Transformations (QVT [25])

standard describes how to provide support to queries and transformations. In contrast to other new languages, QVT uses the pre-existent *Object Constraint Language* (OCL) language to perform queries over software artifacts.

3.1 MOMENT: A framework for Model Management

MOMENT [6] is a tool that provides support to the OMG standards giving capabilities to transform models. The tool uses both an industrial modeling front-end and an algebraic back-end for the execution of the transformation and query tasks. The algebraic background runs in the high performance rewriting system Maude [7]. The industrial modeling environment used by MOMENT is the Eclipse Modeling Framework (EMF). EMF [2] can be considered as an implementation of the MOF standard and can import software artifacts from several heterogeneous data sources: UML models, XML Schemas, etc. With regard to Maude, MOMENT takes advantage of its modularity and parameterization capabilities to provide a metamodel-independent environment to transform models. MOMENT uses the open standard QVT to provide a transformations language in comparison with another popular transformation tools (such as MTF or IBM Model Transformation Framework —MTF [14]— or ATLAS Transformation Language —ATL [3]—) which provide their own propietary languages. The tool offers an implementation of the QVT-Relations language as well as the OCL language. QVT-Relations is a declarative transformations language that provides implicit traceability capabilities. For this language, MOMENT gives wide support for unidirectional transformations. Moreover, the tools provides full support to the query operators of the OCL language.

4 A MDSD approach in biological data migration

In the initial work on the study of the TLR4 signal transduction *pathway*, data migration from the source database to the simulation tool (to represent this information as a coloured Petri net) was done manually.

The solution to the data migration problem is described as follows by means of model transformation techniques using the model-driven software development guides. This implies the following tasks: (a) development of the source domain data model (TRANSPATH®), (b) development of the target domain data model (*CPN Tools*), (c) definition of the transformation rules between the source domain and the target domain by means of the transformations language, (d) implementation of the pre-processing mechanism to obtain the instances of the source model from the original data; and finally, (e) definition of the post-processing tasks that persist the transformed data in the final file format. The next subsections describe the designed solution. First, the transformation process and the different stages are described; second, the source and the target models are presented, and last, the transformation process is explained in more detail.

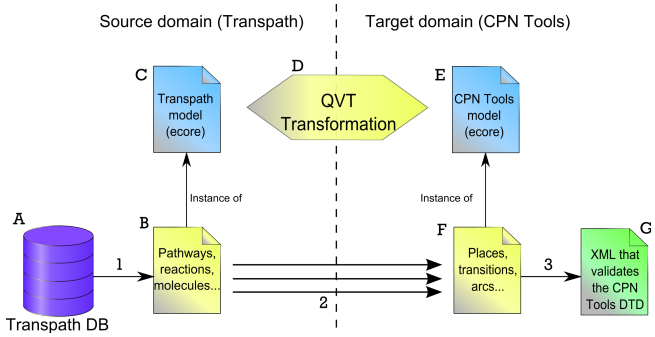

Fig. 4. Architecture of the tool.

4.1 Architecture and overview of the tool

The data migration process is performed in three steps: (1) recovering and pre-processing of the input data, (2) execution of the transformation by means of the transformations engine and (3) post-processing and persistence of the result data. In a MDSD approach, using a transformation engine implies that the source and the target models of the transformation must be developed in the first place to be able to establish the mappings between the two domains.

The solution presented in this paper uses MOMENT which is a tool that is integrated within the Eclipse platform and provides support for model transformations. This tool uses the *Eclipse Modeling Framework* (EMF), which provides Ecore as a modeling language. Moreover, it uses XMI as the persistence format and allows the creation of Ecore models from, among others, XSD Schemas.

Nevertheless, the Ecore models obtained from XSD schemas are complex and do not always clearly represent the real structure of the data. Therefore, the source and the target models have been defined manually, taking into account only the information that is useful in the case study.

Figure 4 shows the architecture of the tool. It represents the three steps that are needed to perform the data migration. First, the data is extracted from the TRANSPATH® database (A), and the corresponding XMI instance (B) of the TRANSPATH® Ecore model (C) is built. This first step (1) is easily done in Java since the mappings between the elements of the source data and the elements of the EMF model can be established directly. The implementation of this pre-processing step has been adapted from the work done in [29].

The second step (2) is the most important and complex one of the transformation process. It is performed by means of the MOMENT tool and its transformation engine. It executes the transformation from the TRANSPATH® domain (reactions, molecules, etc.) to the *CPN Tools* domain (places, transitions, arcs, etc.). After the definition of the transformation rules (D) between the source domain (C) and the target domain (E), the transformation is executed over the data recovered from the database (B) obtaining the needed information in the *CPN Tools* domain (F).

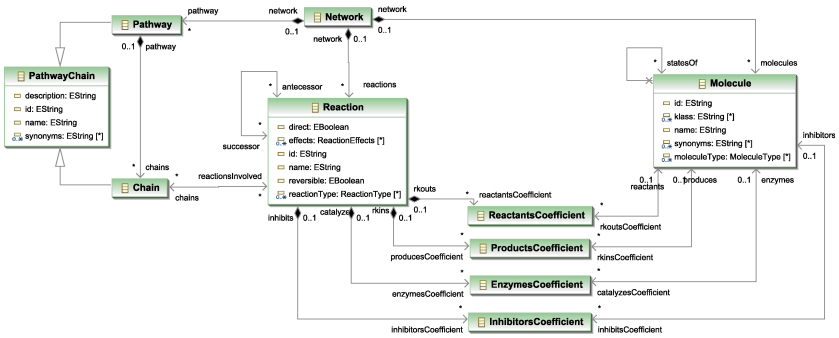

Fig. 5. Model of the TRANSPATH® database.

Finally, the third step (3) in the data migration process is again a trivial process in which the EMF data is stored in an XML file readable from the *CPN Tools* application (G). Other tasks can be performed in this stage; for example, the execution of some layout algorithms over the elements of the Petri net to represent the graphical elements properly in the drawing space of the *CPN Tools* GUI.

4.2 Development of the source and the target models

First, a model that contains the most interesting elements to simulate a signal transduction pathway in the *CPN Tools* application has been defined (removing unnecessary concepts from the complex TRANSPATH® database).

Using a visual metaphor similar to the UML class diagram, Figure 5 shows the Ecore model that has been developed. The *Network* class is the main element in this model. A *Network* contains a set of *Pathways*, *Reactions* and *Molecules*. Moreover, a *Pathway* is composed of several *Chains* of *Reactions*, and one *Reaction* can be involved in several *Chains*. Finally, the reactions are related to the molecules. One molecule can be a reactant or a product in a reaction. It can also take part in a reaction as an inhibitor or a catalyst (if the molecule is an enzyme). The classes *ReactantsCoefficient*, *ProductsCoefficient*, *EnzymesCoefficient* and *InhibitorsCoefficient* inherit from the *Coefficient* class (omitted for reasons of clarity). This class contains an integer attribute (*coefficient*[3]).

Figure 6 shows the model that has been created for the *CPN Tools* application. In this case, the design of the model is closer to the application specific concepts than to the conceptual Petri net concepts. This design allows us to deal with all the interesting concepts of the *CPN Tools* platform (e.g. position and color of the graphical elements). Furthermore, this kind of design makes the persistence process from EMF to the final XML file easier.

[3] The coefficient value represents the number of molecules that appear in the equation of one reaction. For example, in the reaction $2H_2 + O_2 = 2\ H_2O$, the coefficients are the numbers that appear on the left of the molecules, i.e., $\underline{2}H_2 + \underline{1}O_2 = \underline{2}H_2O$

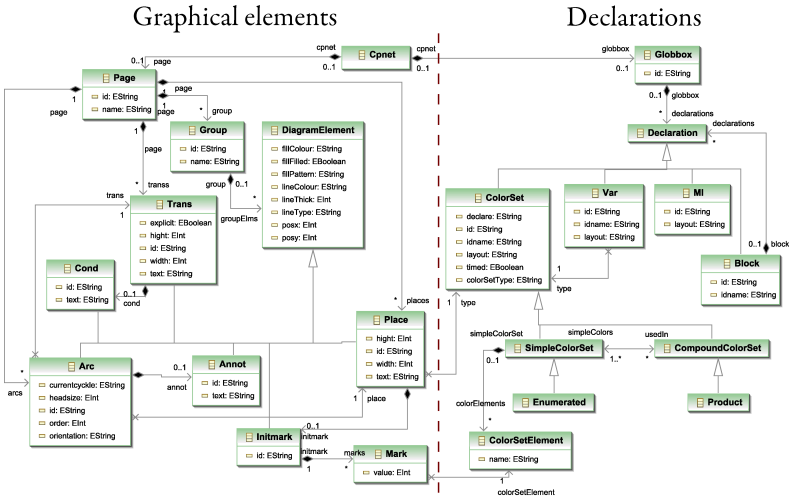

Fig. 6. Model of the *CPN Tools* application.

Cpnet is the main class of the mode (see Figure 6). It is divided, by using a dashed line, into two groups: the classes that are under the *Globbox* element and the classes under the *Page* element. The first group (the *Globbox* group) allows the declarations of CPNs such as colorsets (enumerated, complex), variables, blocks, etc. The second group of classes (those contained in the *Page* element) represents all the visual elements of the coloured Petri net. All the graphical elements inherit from the *DiagramElement* class, and can be contained in different groups (*Group* class). Thus, a *Page* can hold *Places*, *Trans* (transition), *Arcs*, *Annot* (annotations), etc. When a *Place* is defined in the Petri net, it has an associated color set. This color set must be defined previously in the declarations part. The relationship between the *Place* and its color set is represented by means of the *type* role from the class *Place* to the class *ColorSet*. The classes *InitMark* and *Mark* are intended to represent the actual state of a given *Place*, indicating which tokens are in the *Place*. The kind of tokens is defined by the role *colorSetElement* between the classes *Mark* and *ColorSetElement*.

4.3 Transformation process

Finally, the transformation rules that can convert data from the source domain to the target domain have been defined. These rules express the mappings established by biologists between the data extracted from the TRANSPATH® database and the concepts available in the *CPN Tools* application. Table 1 shows the simplified mappings between both the source and the target domain. The rules that define the direct relationships between the two domains have been expressed in QVT-Relations. In this language, a transformation is a set of *relations* established between the domains that participate in that transformations

that must hold for the transformation to be successful [25]. A domain is a typed variable that can be matched in a model to be transformed. A domain also can hold a given *pattern*. This *pattern* can be considered as a set of restrictions that the elements of the *candidate model* (the model to be transformed) must satisfy.

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

Table 1. Mappings between the source and the target domain.

The domains can be characterized by means of the keywords `checkonly` and `enforce` (these keywords are abbreviated as `c` and `e` in the visual representation of the QVT-Relations language). In a `checkonly` domain (or `c` in the visual representation of the QVT-Relations language), the transformation will verify that there is a valid matching (the domain pattern will match) in the candidate model. When a transformation is executed in the direction of the model of an enforced domain, if checking fails, the target model will be modified to satisfy the domain pattern.

The *ReactantsToPlaces relation* (Figure 7) shows an example of the QVT-Relations graphical syntax. It expresses the relationship between the *molecules* and the *places*. The *relation* has two domains: the *tpDomain* corresponds to the TRANSPATH® database, and the *cpnDomain* corresponds to the target domain (*CPN Tools*). Each domain specifies a simple pattern: a molecule with a name, and a place with an id. Both the name and the id property are bound to the same variable *moleculeName1*, implying that they should have the same value.

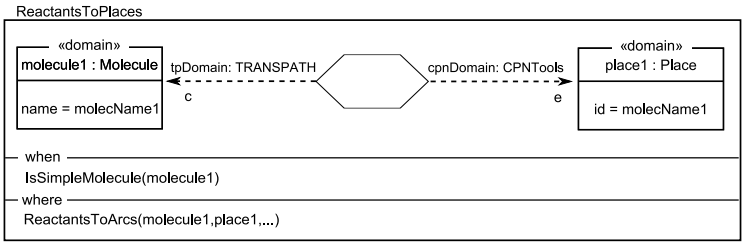

Fig. 7. Relation ReactantsToPlaces

A relation can also be constrained by two sets of predicates, a `when` clause and a `where` clause, as shown in the example relation *ReactantsToPlaces*. The `when` clause specifies the conditions under which the relationship needs to hold. Thus, the *relation* will only be applied if the molecule is simple (IsSimpleMolecule(...) is a function that, by means of an OCL expression, checks if a given molecule is simple or compound). The `where` clause specifies the condition that must be satisfied by all the model elements participating in the relation before the application of the current *relation*.

The transformation is executed as a top-down process. The navigation is performed through the containment relationships (defined in Ecore by means of containment references), i.e., it begins from the root element of the source model (*Network*) and goes down (*Network* → *Pathways* → *Chains* → *Reactions* → *Coefficients* → *Molecules*) creating the corresponding elements in the target domain as Table 1 defines.

In the declarations group, the transformation will create an *Enumerated ColorSet* from each simple molecule. In the case of the complex molecules, the *ColorSet* created will be a *Product*. This *Product* will be be a compound of the *Enumerated ColorSets* corresponding to the simple molecules which are part of the complex molecule.

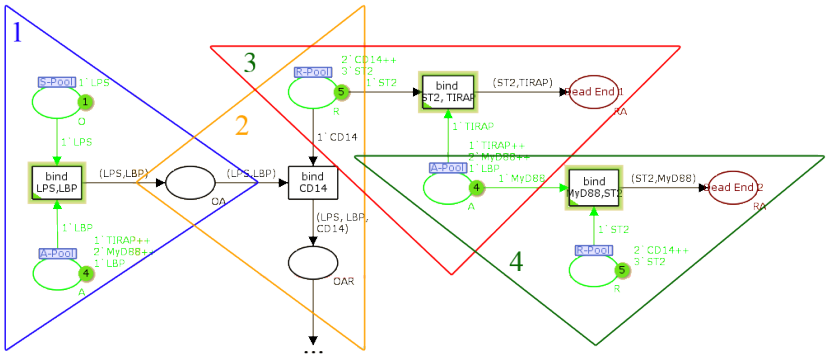

Fig. 8. Partial representation of the TLR4 signal transduction pathway in *CPN Tools*.

In the graphical elements group, the transformation process begins from a *Reaction* element. An object of the class *Trans* is created for each reaction. We obtain the reactant molecules through the *reactantsCoefficient* association in the class *Reaction*. A *place* is created for each one of these molecules. Finally, each new *place* can be linked with its corresponding *Trans* element by means of an *Arc*. These *arcs* will be of type *PtoT* (*Place to Trans*, according to the *CPN Tools* terminology). The procedure is similar for the products of the reactions; however in this case, the transformation navigates through the *productsCoefficient* association.

Figure 8 represents the result of the transformation process (in the *CPN Tools* metaphor) for the reactions presented in the case study. The figure shows four numbered triangles, each of which corresponds to one of the reactions of the example. Thus, for reaction number (1) (LPS + LBP $\leftrightharpoons$ LPS:LBP) the transformation generates the elements inside the left triangle (1). The other three triangles (2, 3, and 4) indicate the corresponding reactions.

5 Conclusions and future work

This paper has presented a case study where the interoperability problem between bioinformatic applications is addressed using a model-driven approach. The situation where several data sources and simulation tools co-exist and must share heterogeneous data is very common in the bioinformatics field. In this situation, the easy representation of biological data using models allows us to deal with these problems more efficiently and more elegantly than the traditional (manual) approaches. It is more efficient because the software development process is shorter. It is more elegant because the operations are done at a higher level of abstraction and the language used is more expressive due to its declarative nature. In [12,5,20,22,27] also model-driven approaches are applied in the life sciences but not in the field of signal transduction pathways.

This work presents the following advantages over the traditional approaches: (1) It allows some tasks that were previously done by hand to be automated. (2) This approach produces more modular tools, making the transformation mechanism independent from the data persistence format, improving the extensibility and maintainability of these tools. (3) Biologists do not need to know technical details about the migration process, which increases their productivity. (4) It also takes advantage of model transformation technologies. Using models to represent the data to be transformed permits the data structure to be more clearly represented making its manipulation more intuitive since it deals with high-level concepts. (5) Traceability capabilities are provided implicitly. These capabilities help to locate invalid information in the data sources. Finally, (6) using languages such as QVT-Relations offers the advantage of expressing the mappings between the source and the target domains in a declarative way. This way of representing the relationships between the two domains is more expressive than the traditional and imperative approaches.

With our case-study we presented the first steps in using model-driven techniques in the live science. Further research are focussed in different goals. First, using models to represent biological data allows us to take advantage of the new *frameworks* such as *GMF*[1] or *MS DSL Tools*. These tools use models to automatically generate visual metaphors. Second, the research done in Model-Driven Engineering can provide a rich background not only for data transformation between two different domains, but also for other tasks such as heterogeneous data integration. Futhermore, we would like to apply this approach to larger and more complicated signaling pathway networks. First comments by experts were positive because the animated models in the CPN Tools reflect the structure of

signal transduction pathways very well and let the biologists watch their pathways being "executed". We would like to integrate biologists deeper in the process of modelling when we start to transform larger signal transduction networks.

6 Acknowledgements

We would like to thank the followinf: C. Choi, E. Shelest, B. Störmann and C. Rio Bartulos for providing us with insights in signal transduction pathways; our colleagues from the Intergenomics project (especially R. Münch) for fruitful discussions; and all the students that contributed through their diploma theses to this project: N. Fleischer, H. Langhorst, M. Pirsch, T. Springmann, M. Ziegler, O. Witthöft, J.-C. Treusch and F. Rudolph. We would also like to thank P. Queralt for his valuable knowledge on the use of the MOMENT-QVT tool.

References

1. Gmf. `http://www.eclipse.org/gmf/`.
2. EMF. `http://www.eclipse.org/emf/`.
3. Atlas transformation language, 2007. `http://www.eclipse.org/m2m/atl/`.
4. B. Alberts, D. Bray, K. Hopkin, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Lehrbuch der Molekularen Zellbiologie*. Wiley-VCH Verlag GmbH / Co. KGaA, Weinheim, 3 edition, 2005.
5. K. Bhattacharya, R. Guttman, K. Lyman, F. Heath, S. Kumaran, P. Nandi, F. Wu, P. Athma, C. Freiberg, L. Johannsen, et al. A model-driven approach to industrializing discovery processes in pharmaceutical research. *IBM Systems Journal*, 44(1):145–162, 2005.
6. A. Boronat, J. Iborra, J. Ángel Carsí, I. Ramos, and 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.
7. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. Maude: specification and programming in rewriting logic. *Theor. Comput. Sci.*, 285(2):187–243, 2002.
8. K. Czarnecki and U. W. Eisenecker. *Generative programming: methods, tools, and applications*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.
9. W. Damm and D. Harel. LSCs: Breathing Life into Message Sequence Charts. *Formal Methods in System Design*, 19(1):45–80, 2001.
10. S. R. S. A. e. a. Emmett, S. Towards 2020 science. Technical report, Microsoft Corporation, 2006. `http://research.microsoft.com/towards2020science/downloads/T2020S_ReportA4.pdf`.
11. J. Fisher, D. Harel, E. Hubbard, N. Piterman, M. Stern, and N. Swerdlin. Combining State-Based and Scenario-Based Approaches in Modeling Biological Systems. LNBI, vol. 3082, pages 236–241, 2004.
12. K. Garwood, C. Garwood, C. Hedeler, T. Griffiths, N. Swainston, S. Oliver, and N. Paton. Model-driven user interfaces for bioinformatics data resources: regenerating the wheel as an alternative to reinventing it. *BMC Bioinformatics*, 7(1):532, 2006.

13. M. Hucka, A. Finney, B. Bornstein, S. Keating, B. Shapiro, J. Matthews, B. Kovitz, M. Schilstra, A. Funahashi, J. Doyle, and H. Kitano. Evolving a Lingua Franca and Associated Software Infrastructure for Computational Systems Biology: The Systems Biology Markup Language (SBML) Project. *Systems Biology*, 1(1):41–53, June 2004.
14. C. IBM. The model transformation framework web site.
15. K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*. Springer, Berlin, 2nd edition, 1997.
16. K. Jensen, L. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *Int. J. on Software Tools for Technology Transfer (STTT)*, Sp. Sec. CPN 04/05, 2007.
17. G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. R. Gopinath, G. R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Research*, 33(suppl_1):D428–432, 2005.
18. M. Kanehisa, S. Goto, M. Hattori, K. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Research*, 34(suppl_1), 2006.
19. P. Karp, C. Ouzounis, C. Moore-Kochlacs, L. Goldovsky, P. Kaipa, D. Ahren, S. Tsoka, N. Darzentas, V. Kunin, and N. Lopez-Bigas. Expansion of the BioCyc collection of pathway/genome databases to 160 genomes. *Nucleic Acids Research*, 33(19):6083–6089, 2005.
20. G. Komatsoulis, D. Warzel, F. Hartel, K. Shanbhag, R. Chilukuri, G. Fragoso, S. Coronado, D. Reeves, J. Hadfield, C. Ludet, et al. caCORE version 3: Implementation of a model driven, service-oriented architecture for semantic interoperability. *J Biomed Inform*, 2007.
21. M. Krull, S. Pistor, N. Voss, A. Kel, I. Reuter, D. Kronenberg, H. Michael, K. Schwarzer, A. Potapov, C. Choi, O. Kel-Margoulis, and E. Wingender. TRANSPATH(R): an information resource for storing and visualizing signaling pathways and their pathological aberrations. *Nucleic Acids Research*, 34(suppl_1):D546–551, 2006.
22. H. Li, J. Gennari, J. Brinkley, et al. Model Driven Laboratory Information Management Systems. *AMIA Annu Symp Proc*, 484:8, 2006.
23. S. Motta and V. Brusic. Mathematical Modelling of the Immune System. Modelling in Molecular Biology, pages 193–218. Springer-Verlag Berlin, 2004.
24. Object Management Group. MDA Guide Version 1.0.1. 2003. `http://www.omg.org/docs/omg/03-06-01.pdf`.
25. Object Management Group. MOF 2.0 QVT final adopted specification (ptc/05-11-01). 2005. `http://www.omg.org/cgi-bin/doc?ptc/2005-11-01`.
26. Object Management Group. Meta Object Facility (MOF) 2.0 Core Specification (ptc/06-01-01), 2006. `http://www.omg.org/cgi-bin/doc?formal/2006-01-01`.
27. Y. Song, E. Kawas, B. Good, M. Wilkinson, and S. Tebbutt. DataBiNS: a BioMoby-based data-mining workflow for biological pathways and non-synonymous SNPs. *Bioinformatics*, 23(6):780, 2007.
28. C. Taubner, B. Mathiak, A. Kupfer, N. Fleischer, and S. Eckstein. Modelling and simulation of the TLR4 pathway with coloured petri nets. *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, pages 2009–2012, August 2006.
29. M. Ziegler. Implementierung eines Transformationsmoduls in Java zur Abbildung von Signaltransduktions-Instanzen auf CPN-Instanzen. Master's thesis, Technische Universität Braunschweig, Braunschweig, Januar 2007.