METAMODEL AND MODELING LANGUAGE: TOWARDS AN UNIFIED MODELING LANGUAGE (UML) PROFILE FOR SYSTEMS BIOLOGY

M. ROUX-ROUQUIÉ*, N. CARITEY, L. GAUBERT, B. LE GRAND, M. SOTO
LIP6 CNRS-UPMC 8 rue du Capitaine Scott, 75015 Paris, France

ABSTRACT

Since data integration is a first-order priority in systems biology, metamodeling that is the foundation for data integration, should become an effective strategy among current approaches. In practice, the building of a metamodel requires four levels: the information level 1 or data level, which consists in the basic facts to integrate, the data model at level 2, i. e. how the data are organized, the metamodel (level 3) that describes and organizes concepts with a set of well-formed rules to integrate models from level 2 and at level 4, the language for metamodeling that uses concepts and relations defined in the metamodel.

Based on these principles, we developed an object-oriented systemic metamodel understandable by humans and computers that makes it possible to integrate data on structure, on the one hand and function, on the other hand. This metamodel was grounded on the active class, *FunctionalUnit*, whose attributes specify time-, space- and shape-dependency of biological functions. A special emphasize was put on the "shape" attribute, named *FormOccurrence*, to account for the multifarious functional isoforms with respect to the gene or the protein from which they are derived. The context-dependency of biological functions was assessed through specific roles devoted to the *FunctionalUnit* class, they are vicinity (*NeighborFunctionalUnit*) for local context-dependency and globality (*SupraFunctional Unit*) for global context-dependency.

We used the Unified Modeling Language (UML) to design our metamodel and we present arguments for customizing it to systems biology.

Keywords: metamodel, data integration, systems biology, UML profile.

1. WHAT IS A METAMODEL AND WHAT IS IT NEEDED FOR ?

A metamodel is a conceptual framework made of rules and elements that allows specifying any kind of models, this model of models containing all concepts and relations present in the subsumed models [1]. This approach requires a sharable understanding of the term of "model" as various definitions exist according to the scientific community. In biology, a model is a verbal description accompanied, when possible, with a graphical representation; in order to introduce standardized elements in these practices, a graphical language has been developed recently [2]. In the database community, models refer to data schemas on which the databases are developed [3]; the current paradigms for modeling data being the Entity-relation (ER) paradigm and Object-Oriented (OO) one. In mathematics, models are analytical representations using a variety of formalisms, for example the ordinary differential equations (ODEs) or the Boolean networks, each of them being able to capture some special features as continuous or discrete properties, respectively [4]. Although very different, these meanings result each time from an abstraction of the real world through the identification of relevant elements. A model upwards these models that would contain all elements and rules organized into a coherent conceptual framework consists in a metamodel. Moreover, a metamodel captures the syntax and the semantics of models contained in it; as such, it allows defining a real language at the meta-meta level. For example, "classes" and "relations" in the object-oriented UML language are the language elements that are used for tailoring UML to a specific application domain; this tailoring is named "profiling" and results in a domain-specific language.

Metamodeling is specially needed when systems to be modeled are so complex that they cannot be represented in a single model and multiple models have to be subsumed to give a relevant understanding of the complete systems. Otherwise, when different levels of abstraction are required like molecular and clinical in trial series, metamodeling could allow these models to coexist for generating new knowledge. Because model syntax is fitting with metamodeling, metamodeling can be used in model transformation, a model developed in one formalism being transformed into another formalism.

Systems biology, which is an emerging field dedicated to the understanding of the functioning of living systems through the study of the complete set of their components (genome, transcriptome, proteome, etc.), is facing most of these challenging issues that metamodeling aims to decipher. The present work describes the design and the instantiation of a systemic metamodel (SMM) in systems biology.

2. A SYSTEMIC METAMODEL IN SYSTEMS BIOLOGY.

The general scope of systemic approaches is to assess the functioning and the evolution of any entity considered as a system. As reported previously [5], the systemic principles state that:

1. An entity taken as a system can be efficiently represented as the interface between an internal and an external environment in which it is evolving and on which it is acting,
2. Its behavior is described as the state trajectory in a time, space, form frame. Events occurring from internal and/or external environment allow changes in state variables and consecutive firing of state transitions.

From an operational point of view, the adaptation of systemic principles to systems biology provides a SMM framework for guiding the integrative annotation of genome, transcriptome, proteome, etc., and linking data on structure, on the one hand, to data on functioning, on the other hand. The SMM design was achieved using the object-oriented UML language that fits in the main systemic principles [5].

*to whom correspondence should be addressed : magali.roux@lip6.fr
Tel (+33) 144 277 094 / Fax (+33) 144 277 495

The structure level: data on substance

First of all, a SMM should make it possible to identify the list of all biological structures as the elementary biological element, or "substance", at a special level, *i.e.* a molecule at the molecular level, a cell at the cellular level, *etc*. The substance concept specifies elements which are context-independent and have unchanged attribute values over time; this must be clearly distinguished from the set of states taken by the corresponding entity and that refers to the entity history.

In object-oriented modeling, biological substances are described as a *is-a* and *has-a* hierarchies. Many works have been involved with classification of biological concepts; the ontology in figure 1 presents a partial hierarchical organization of biological substances from molecules to organisms, the prefix "S" indicating that the specific class is derived from the *Substance* upper class. Accordingly, a protein *is-a* molecular substance (*S_Molecule*) that *is-a* substance, a nucleus membrane is-a organelle substance (*S_Organelle*) that also *is-a* substance. A lymphocyte *is-a* cell substance (*S_Cell*) that *is-a* substance as well. Otherwise, a *S_Cell* (lymphocyte) *has-a* *S_Organelle* (nucleus membrane) that *has-a* *S_Molecule*. This hierarchy takes advantage of object-oriented features, inheritance, on the one hand, which allows an object to inherit properties from the object from which it is derived and modularity, on the other hand, that consists in breaking up complex entities into manageable pieces or modules. Several principles govern modularity, notably understandability (a module must be fully understandable without referring to another module) and composability (the elements can be combined to produce new systems); with this respect, the *Substance* hierarchy realizes a *Substance* module.

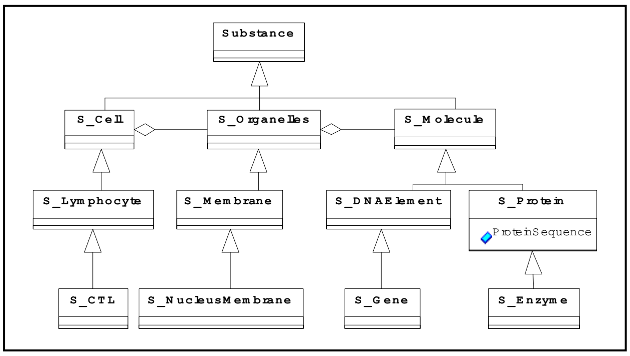


Figure 1. The *Substance* hierarchy (see teext for details).

To illustrate the context-independent features of the Subtance class, some attributes of the gene taken as a substance are presented in Table 1: A gene substance has a unique identifier, with an approved gene symbol (for example, human genes) and some additional symbols. According to the approved gene symbol, the gene has a chromosomal location and a nucleotide sequence which are species-specific; this DNA sequence is coding for RNA and protein sequences.

Table 1. Attributes of the *GeneSubstance* class

MD_geneId: ID	Accession numbers to any other database
GeneSymbol: String	Usual name
OtherSymbol: String	Other names
ChromosomalLocation: String	MD_GeneCards database Identifiers
S_promoter: S_Promoter	Gene promoter
S_regulatoryElement:S_regulatoryElement	Gene regulatory elements
S_rna: S_RNA	Encoded RNA(s)
S_protein : S_Protein	Encoded protein(s)
startPosition : int	An integer indicating starting position; this position is given according to the initiation site.
endPosition : int	An integer indicating ending position; this position is given according to the initiation site.
…	

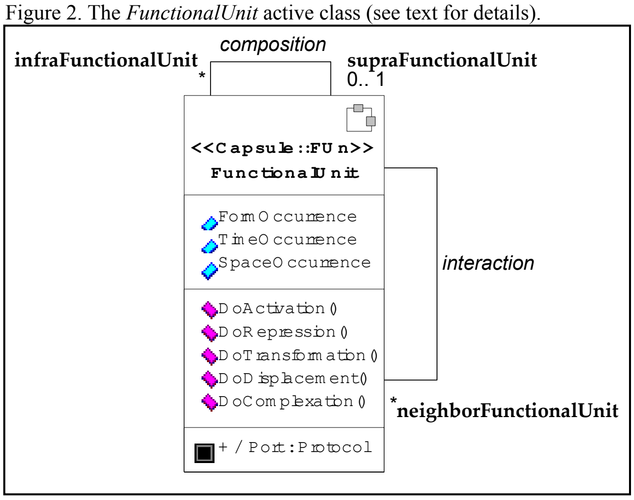
Accordingly, the Substance package integrates unique invariant data on biological entities as unique identifier and name; this is a pivot package ground on gene or protein Ids that enable access to diverse annotation sources. The merging of transcriptome, proteome and metabolome data models has been realized recently, it integrates data from each source into a unique resulting SysBio-OM model [6]. In our hands, the integration of the SysBio-OM model into SMM should occur through the BioSequence package associated to the BioMaterial package and especially to the specialized SpotMaterial, PeakMaterial and ColumnfractionMaterial. Ontology-based strategy has been proposed for insuring semantic integration of databases [7]. We think that *Substance* ontology in SMM would provide the first clue to integrate the main annotation ressources.

The function level: data on dynamics

The SMM has to identify the list of functions of the system components. This is a difficult task as function is very versatile due to time-, space-, shape- and/or context-dependency. The Gene Ontology (GO) [8] has developped a controlled vocabulary that distinguish "molecular function" and "cellular process" (in addition to the "compartment" hierarchy), which is of unvaluable help to improve standardization in biological annotation; nevertheless, this approach does not allow a dynamic description of biological behavior in terms of inputs and outputs to "function" from one state to another state through transition firing. Integrating data on dynamics requires to deal explicitly with context in a space-time-shape frame as biological functions differ accounting to the context. Think to different Wnt signalling outcomes in early embryo (axis specification, organogenesis, …) and adult cell (bone density, cancer, …) Similarly, the function of some proteins differ according to the 3D-shape (for example, the phosphorylation of the Cdc2 kinase on T14 and Y15 induces a conformational change that inhibits its enzymatic activity). We account for these dynamic features by introducing the concept of "functional unit" distinct from the concept of "substance" and that is defined according to the systemic guidelines; *i. e.* as an active component in an active environment in which it functions and transforms itself [5].

In the object-oriented paradigm, the *FunctionalUnit* (FUn) class is an "active" class, *i. e.* a class , which has its own thread of control (designed as a "capsule" in figure 2; Rose Real-Time, IBM). In addition to having attributes and operations as passive classes, active classes have "ports" that are interfaces for communication; the protocol for communication being specified in a special "protocol" class. Figure 2 presents the SMM core consisting in the active *FunctionalUnit* class and relations.

Figure 2. The *FunctionalUnit* active class (see text for details).



1. The *FUn* entity is the elementary unit in a process, it can be tangible (a functional molecule) or intangible (a functional network), The prefix "FUn" is used to refer to classes or objects that are derived from the *FUn* upper class.

2. FUn roles are SupraFunctionlUnit, InfraFunctionalUnit and NeighborFunctionalUnit: A *SupraFunctionlUnit* (*SupraFUn*) corresponds to the *external environment* or "global context" environment for a specific *FUn* (for example, an epithelial cell will be the external environment (*SupraFUn)* of the *FUn_protein*, for example, EGF receptor. Accordingly, a *FUn* may have 0 (unknown) to 1 *SupraFUn*. Conversely, the *internal environment* would be assessed by *InfraFunctionalUnit (InfraFUn) FUn_protein* and *FUn_DNA* would behave as *InfraFun* according to the *FUn_nucleus* in which they are embedded. Cardinality for *InfraFun* would range from 0 to n. Composition relationships are linking *InfraFuns* to *FUns* to *SupraFUns*. The *local environment* or "local context" of *FUns* is made of *NeighborFunctionalUnits (NeighborFUns)* with which *FUns* have *interaction* relationships. Thes relations may be of different kinds depending of the abstraction level for FUns (from physical molecular complexation to abstract genetic relations).

FUn attributes are *FormOccurrence*, *TimeOccurrence* and *SpaceOccurrence*, according to systemic principles.

3. The *FormOccurrence* describes the special details of the shape involved in the actual activity of the *FUn*. This description can be a string if the *FUn* is a sequence, it will be quantitative in cells taken as *FUns,* and equations will define parts of the Euclidian space. The *FormOccurrence* is a composite attribute composed of Substance and Transformation (Figure 3).

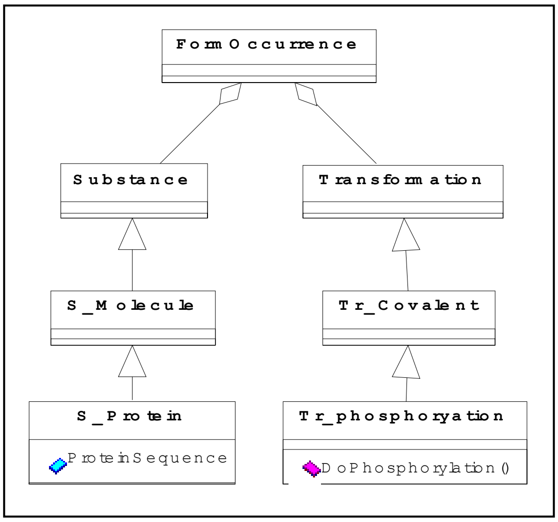


Figure 3. The *FormOccurrence* composite attribute (see text foe detains)

4. The *Transformation* class specifies a transformation that occurs to modify the form of the *FUn*, it has the operation *DoTransformation()* and every derived class will inherit of this operation although the associated method will be specific to each particular child class. At the molecular level, transformations distinguish non covalent transformations (*Tr_NonCovalent*) and covalent transformations (*Tr_covalent transformation)*, the prefix *Tr_* being used to name classes or objects derived from the *Transformation* class. In the example presented in Figure 3, the *FormOccurrence* of a phosphorylated FUn_protein is updated following a *DoPhosphorylation()* operation call, pAAc being the amino acid position :

getProteinSequence(p : S_Protein)
phosphorylates(b : ProteinSequence, pAAc : int)

5. The *TimeOccurrence* describes the moment, the instant during which the *FUn* is acting. It is used to specify the age of the entity and refers to an internal clock, or it delineates the period of time the FUn is acting.

6. The *SpaceOccurrence* describes the Fun's position according to the external environment. The *SpaceOccurrence* refers to the *formOccurrence* of the containing FUn.

7. FUn operations are:
   * DoDisplacement () that updates the spaceOccurrence attribute
   * DoTransformation () that updates the formOccurrence attribute
   * DoActivation () that changes the current state of the target
   * DoInhibition () that changes the current state of the target

The operation calls are realized on signal entrance through ports.

8. FUn ports are special devices of classes to manage in/out signals. They have attached protocols that specify that messages exchanged between two objects are conform to some interaction pattern. For example, the *Phosphorylation* protocol class will exchange messages *DoPhosphorylation* and *Phosphorylated* between the kinase and the target (see Figure 4). A sequence diagram specifies the interaction between objects; it shows the explicit ordering of message passing over time (not shown).

3. INTEGRATING DATA ON STRUCTURE AND BEHAVIOR: THE MPF CASE STUDY.

A case study was selected to illustrate the coupling between structural and behavioral data; this concerns the G2/M transition in cell cycle, which is triggered by the Cdc25-mediated activation (dephosphorylation) of the cyclinB/Cdc2 complex (MPF) involved in the regulation of events receding cell division. At the S/G2, the complex is kept in cytoplasm in an inactive state due to phosphorylation of T14 and Y15 on Cdc2, catalyzed by Wee1 and Myt1 kinase, respectively. The complex is further activated by phosphorylation of T160 and the dephosphorylation of T14 and Y15 on Cdc2. We concentrated on the negative regulation of MPF complex to show how structure and behavior data can be integrated into a unique model.

The following FUns entities were created: *FUn_Cdc2*, *_CyclinB*, *_MPF*, *_Myt1* and *_Wee1*; let consider data about *FUn_Cdc2*:

At the structure level, *S_Cdc2* has approved gene symbol CDC2, synonym symbol CDK1 and chromosomal location on 10q1.1; in addition, the DNA sequence refers to NT 008583 contig of Genbank and protein sequence has Uniprot/Swiss-Prot identifier: CDC2_HUMAN, P06493

At the behavioral level, *FUn_Cdc2* TimeOccurrence, SpaceOccurrence and FormOccurrence attributes have value: S and/or G2, S_cytoplasm and S_Cdc2 protein sequence, respectively. Its global environment is the *SupraFUn*: *FUn_cytoplasm*, and the aggregation relation to *FUn_CyclinB* creates *FUn_MPF*.

Figure 4a presents *FUn_Cdc2* in association with *FUn_CyclinB* to compose *FUn_MPF*. *FUn_Cdc2* is phosphorylated by messages passing between *FUn_Cdc2*, *FUn_Myt1* and *FUn_Wee1* according to the protocol *Phosphorylation*. Ports on *FUn_Cdc2* are *T14* and *Y15* and *KinaseDomain* on *FUn_Myt1* and *FUn_Wee1;* protocol *Phosphorylation* has InSignal: *DoPhosphorylation* and OutSignal: *Phosphorylated*.

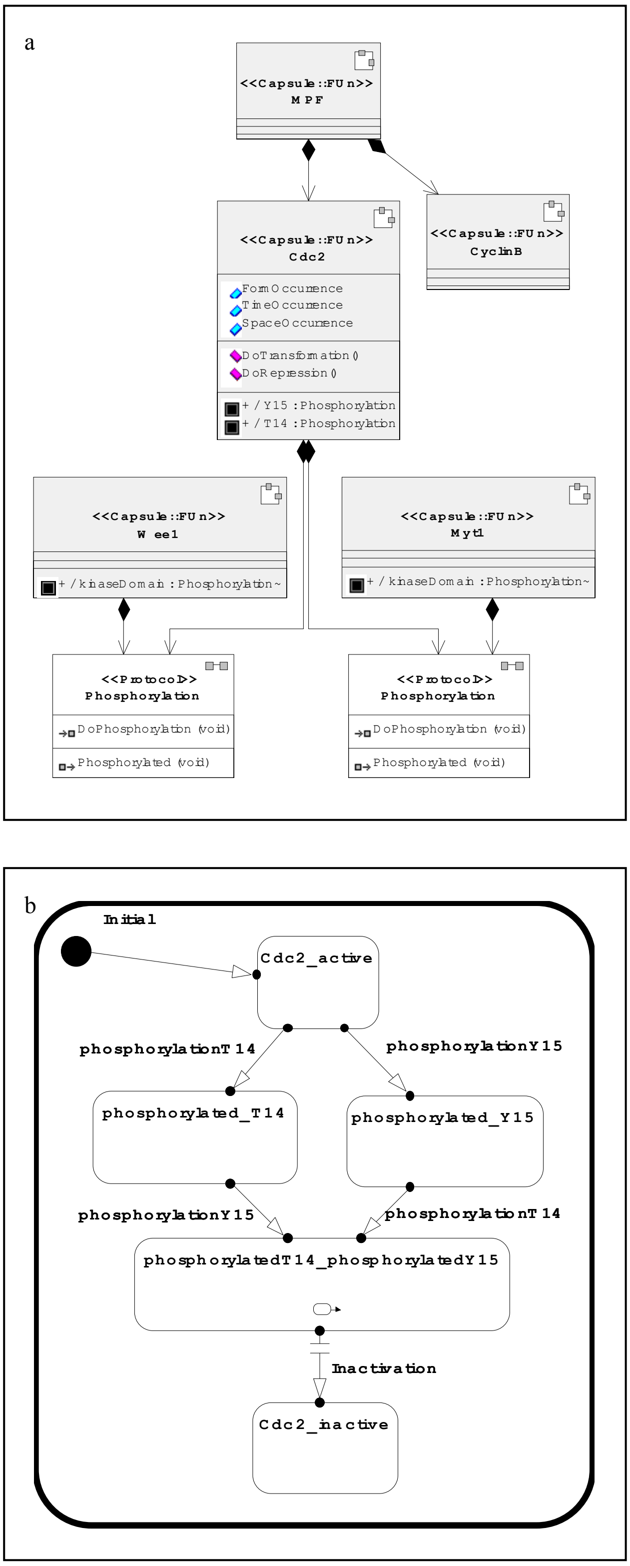


Figure 4. Phosphorylation of the Cdc2_CyclinB_complex using active classes and protocols (a); States and state transitions of the FUn Cdc2 (b) (see text for details).

Figure 4b presents the *FUn_Cdc2* state diagram: Cdc2 in the active state is phosphorylated on T14 (left), the transition firing was achieved as the port T14 received the message *DoPhosphorylation* wih the guard condition *Myt1 True* that updates Cdc2 *FormOccurrence* to *T14phosphorylated sequence.* Next, port *Y15* receives the entering message *DoPhosphorylation* and transition is fired to the new state with *FormOccurrence* value: *T14Y15phosphorylated sequence; t*he opposite scenario is performed on right lane. At the exit of the state *phosphorylatedT14phosphorylatedY15*, an operation call *DoInactivation*, is firing the transition to a new *inactivate* state.

4. DISCUSSION

In "The Molecular Biology Database Collection: 2005 update" [9], Michael Galperin identifies 719 databases, that is 171 more than the 2004 issue. This 30% increase illustrates how information is important in modern biology and the challenge it is facing for integrating new data and knowledge. Recently, Lincoln Stein reviewed integration approaches [10], and main strategies concern "view integration" and "Data warehousing", both of which have been temptatively developed and failed to be adopted by the community because of the difficulty of maintaining these tools when too many changes are occurring in data models. With this respect, the author advocates sociological problems instead of technical problems, as data providers would need to wonder about how their data will be used. In our opinion, problems are not only technological and/or sociological but also biological: what do we need to know? In what context? What do we need not to know? Etc. Stein proposed a new approach to answer these questions that he named "the knuckles-and-nodes approach" and that consists in facing a specific biological problem, for example "orthology" or "citation" and having data and knowledge curated by scientific groups to insure links with relevant databases. Unless such organization is achieved, these efforts would need to be integrated in a larger biological framework as systems biology aims to realize. Such a framework approach is under development at the CCPN project [11, 12], which consists in several packages, for example, the NMR package depending on the Molecule package that contains descriptions of sequence and covalent geometry.

In the present paper, this framework approach was extended to metamodeling with the design of a systemic metamodel that allows coupling and integrating data on structure and behavior. This was performed by taking advantage of UML elements (class diagrams, sequence diagrams, state diagrams, active and passive classes, etc) to achieve models understandable by humans and computers. With respect to the CCPN descriptions of sequence and covalent geometry, these data can be integrated in our *Substance* and *Transformation* hierarchies, respectively; in addition, the behavior of the corresponding functional molecules can be described in our systemic metamodel using state diagrams; for example, dynamic descriptions of protein folding or docking would make it possible to further simulate molecular space-time evolution, then to couple data to simulation.

One of the goals of the metamodel described above is to take into account all biological factors –static as well as dynamic -, at various scales. The next step is to instantiate this metamodel to perform "in silico" experimentations. As this is the case with traditionnal types of experimentations, the results will need to be analyzed in order to interpret them and ultimately predict the evolution of the system. The volume and multidimensionnality of the generated data will require the use of knowledge extraction techniques such as factor- or conceptual-analysis.

Modeling complex systems, such as biological components, is an error prone process. The validity of the model is thus a central question particulary when this model is used for building in silico experimentations. Mainly two reasons can lead to an invalid model : input data or parameters and the model itself. In the later case, using state machines facilitate the necessary model checking. Indeed, most model checking tools have their own formal language for defining models, but most of them are variant state machine. Modeling is also a highly time consuming process. Model reusability is thus a strategic

property to reduce cost and time in building new models. Using an object-oriented systemic metamodel increases reusability property of instanciated models but the use of modeling language base on state machine allows to partialy automate the adaptation of models to be reused or coupled.
We selected UML because it is widely used for modeling and has special procedures for automatic translation to XML. In addition, UML Real-Time (UML-RT) extensions have a object-constraint language (OCL) allowing semi-formal descriptions of states and state transitions as presented. The use of UML for specifying biological systems would allow tailoring it to systems biology as previously suggested [13]. This would greatly benefit the method developped recently to reverse engeneered biological models intoobject-oriented software systems [14].

References

[1] M. Roux-Rouquié, M. Soto, "Virtualization in systems biology : Metamodel and modeling language for semantic data integration", **Transactions in Computational Systems Biology**, 2005, in press.

[2] H. Kitano, "A Graphical Notation for Biological Networks", **Biosilico**, Vol. 1, 2003, pp. 169-176.

[3] U. Wittig, A. De Beuckelae, "Analysis and comparison of metabolic pathway databases", **Briefings in Bioinformatic**s, Vol. 2, 2001, pp. 126_142.

[4] H. De Jong, "Modeling and simulation of genetic regulatory systems: a literature review" **J. Comput. Biol**., Vol. 9, 2002, pp. 69–105.

[5] M. Roux-Rouquié, J-L. Le Moigne, "The systemic paradigm and its relevance to the modelling of biological functions", **C. R. Biol**., Vol. 325, No 4, 2002, pp. 419-430.

[6] S. Xirasagar, S. Gustafson, B. A. Merrick, K. B. Tomer, S. Stasiewicz, D. D. Chan, Kenneth J. Yost, J. R. Yates, S. Sumner, N. Xiao, M. D. Waters, "CEBS object model for systems biology data, SysBio-OM", **Bioinformatics**, Vol. 20, 2004, pp. 2004-2015.

[7] J. Köhler, S. Philipp, M. Lange, "SEMEDA: ontology based semantic integration of biological ressources" **Bionformatics**, Vol. 19, 2003, pp. 2420-2427.

[8] M. A. Harris, J . Clark, A. Ireland, J. Lomax, M. Ashburner, R. Foulger, K. Eilbeck, S. Lewis, B. Marshall, C. Mungall, J .Richter, G. M. Rubin, J. A. Blake, C . Bult, M . Dolan, H . Drabkin, J. T. Eppig, D. P. Hill, L. Ni, M . Ringwald, R. Balakrishnan, J. M. Cherry, K. R. Christie, M. C. Costanzo, S. S. Dwight, S. Engel, D. G. Fisk, J. E. Hirschman, E. L. Hong, R. S. Nash, A. Sethuraman, C. L. Theesfeld, D. Botstein, K. Dolinski, B. Feierbach, T. Berardini, S. Mundodi, S. Y. Rhee, R. Apweiler, D. Barrell, E. Camon, E. Dimmer, V. Lee, R. Chisholm, P. Gaudet, W. Kibbe, R. Kishore, E. M. Schwarz, P. Sternberg, M. Gwinn, L. Hannick, J. Wortman, M. Berriman, V. Wood, N. de la Cruz, P. Tonellato, P. Jaiswal, T. Seigfried, R. White, Gene Ontology Consortium, "The Gene Ontology (GO) database and informatics resource", **Nucleic Acids Res.**, Vol. 32 Database issue, 2004, pp. D258-61.

[9] M. Y. Galperin, "The Molecular Biology Database Collection: 2005 update", **Nucleic Acids Res**., Vol. 33 Database Issue, 2005, pp. D5-24.

[10] M. Stein, D. Lincoln, "Integrating biological databases", **Nature Genetics** 4, 2003, pp.337-345.

[11] R. H. Fogh, W . Boucher, W. F. Vranken, A. Pajon, T. J. Stevens, T.N. Bhat, J. Westbrook, J. M. Ionides, E. D. Laue, "A framework for scientific data modeling and automated software development", **Bioinformatics**, 2004 Dec 21.

[12] A. Pajon, J. Ionides, J. Diprose, J. Fillon, R. Fogh, A. W. Ashton, H. Berman, W. Boucher, M. Cygler, E. Deleury, R. Esnouf, J. Janin, R. Kim, I. Krimm, C. L. Lawson, E. Oeuillet, A. Poupon, S. Raymond, T. Stevens, H. van Tilbeurgh, J. Westbrook, P. Wood, E. Ulrich, W. Vranken, L. Xueli, E. Laue, D. I. Stuart, K. Henrick, "Design of a data model for developing laboratory information management and analysis systems for protein production ", **Proteins**, Vol. 58 No 2, 2005, pp. 278-84.

[13] M. Roux-Rouquie, N. Caritey, L. Gaubert, C. Rosenthal-Sabroux, "Using the Unified Modelling Language (UML) to guide the systemic description of biological processes and systems", **Biosystems**, Vol. 75, 2004, pp. 3-14.

[14] D Shegogue, W. J. Zheng, "Object-oriented biological system integration: a SARS coronavirus example", **Bioinformatics**, 2005, Feb 24.