Considerations of graph-based concepts to manage of computational biology models and associated simulations

Ron Henkel[1], Nicolas Le Novère[2], Olaf Wolkenhauer[1,3], Dagmar Waltemath[1]

[1] Junior Research Group SEMS, Department for Systems Biology and Bioinformatics, The University of Rostock, D-18057 Rostock, Germany
[2] EMBL - European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridgeshire CB10 1SD, United-Kingdom
[3] Stellenbosch Institute for Advanced Study (STIAS), Wallenberg Research Centre at Stellenbosch University, Marais Street, Stellenbosch 7600, South Africa

[ron.henkel | dagmar.waltemath | olaf.wolkenhauer]@uni-rostock.de, lenov@ebi.ac.uk

Abstract: Over the past years various databases in Life Sciences have been developed, among them databases to handle computational models of biological systems. Exchange formats that represent these models are typically XML-based; they encode models as networks. Models are published together with supplementary materials such as annotations, simulation experiment descriptions, or result sets. In consequence, not only model files need to be managed, but also the associated simulation setups, and highly linked meta-information. We discuss here the use of graph databases for model storage as they well reflect this interrelated nature. They can also enhance the integrated management of computational models and associated meta-information, as they handle connections between models, simulation descriptions and result data files, as well as external knowledge. This property enhances version control, retrieval and ranking, thereby resulting in improved model reuse and result reproducibility.

Keywords: computational biology model, standards, XML, graph database

1 Background

Systems Biology is the study of complex biological systems by means of computational approaches and methods. A computational model of a biological system represents aspects of that system, often using mathematical equations. Modeling has become a major tool in the daily work of systems biologists. Consequently, the number of available models has grown steadily over the last decade, and so has the models' complexity [He10]. To reuse existing model code, the code itself must, first, be made available in model databases. Second, it must be encoded in exchangeable standard formats, which can then be interpreted by software tools. BioModels Database [Li10] is one example of an open model repository that freely distributes models in standard format. Together with the model, a whole plethora of meta-information is provided,

including the reference publication, the model authors, the semantics of the encoded entities, the model curation state, the underlying mathematics, or the graphical representation of the model in a network structure. One particularly important type of meta-data are annotations which provide links from model entities into bio-ontologies [BS06] (e.g., Gene Ontology, GO [As00]). The model code is typically encoded in so-called model representation formats. Frequently used formats are all XML-based; examples are the Systems Biology Markup Language (SBML [Hu10]), CellML [Cu03], or NeuroML [Gl10]. These formats encode in the XML structure the necessary information to rebuild the model structure and underlying mechanisms in a software environment, e.g. for simulation studies. One distributor of SBML models is BioModels Database which currently contains 421 curated models and several thousands of automatically generated pathway models; the majority of models are concerned with signal transduction and metabolic processes[1]. Current model databases consider model code as a unitary entity while model meta-data and meta-information are extracted from the model code and being stored separately, as are supplementary materials. For example, changes in the model code are tracked by a version control system outside the databases (e.g., using SVN or Mercurial). We would here like to point out that this procedure cannot guarantee consistency for files associated to a model. In particular, changes applied to the model are often unnoticed, and their consequences on other files (e.g., linked models or simulation setups reusing the code) are not foreseeable. On the contrary, managing the model files in a database would allow for flexible linkage of files that depend on the model code or on each other (e.g., different versions of a model). Current databases also build on relational storage approaches. This architectural choice dates back to times when standardization only began and model files were not yet associated with such a rich set of meta-data. Since then the databases have grown and functionalities have been extended, but few systems' architectures have been revised. The great amount of meta-information associated with today's models, and the fact that models represent network structures led us to argue for a graph-based storage solution integrated in a model management system rather than a relational approach. NoSQL approaches together with semantic web applications already gained popularity in Life Sciences [Sp11], e.g. Key-Value Stores, BigTable (based on the Google File System [GGL03]), document databases, or graph databases [AG08]. A network-structural approach to data storage has been successfully applied in projects like Bio4J[2], a graph-based database for bio-ontologies (e.g., Uniprot or GO). We will here focus on the graph database Neo4J [Vi10]. It is based on the concept of describing data in terms of nodes, relations and attributes. Nodes are connected via directed relations of certain types. Both, nodes and relations can then hold attributes. The Neo4J architecture follows the fundamental properties of databases, i.e. the ACID principles (atomicity, consistency, isolation, durability).

Among other entities, an SBML-encoded model often represents a biological system as a collection of inter-related compartments, species and reactions: A compartment is a bounded space that can contain species; a species is contained in a com-

[1] http://www.ebi.ac.uk/biomodels-main/

[2] http://www.bio4j.com/

partment and takes part in a reaction; a reaction has reactants, products and modifiers (e.g., to model enzymatic reactions); all are represented by species. The result is a reaction network. In the case of SBML, annotations are linked using RDF [La98]. The RDF triplet relates a model constituent to an entry of an ontology or controlled vocabulary. In consequence, the link between a model constituent and its corresponding entries in ontologies also represent part of the network spanned by the model. Finally, the entries in the ontologies are interconnected thus forming a network themselves. The consequential question is: If SBML models encode networks - why do not we store them as graphs rather than using a relational approach?

2 Storage concept

We will elucidate the storage approach with an example toy model representing the well-known Michaelis Menten kinetics for the formation of an enzyme-substrate complex. For reasons of simplification, we will not model the ES complex but go directly from E + S to E + P instead (while adapting the ratio). To encode this model in SBML, we define three species (E, S and P) and one reaction (R). All entities are being defined in a bounded space, a compartment C. In the reaction R, species E acts as a modifier, species S acts as a reactant and species P acts as a product. The SBML code for the above kinetics is given in Figure 1. In a complete example, all defined model entities should be annotated, for example, with entries from the Systems Biology Ontology for terms commonly used in computational modeling [Co11], or GO. Finally, for the model itself, the authors and the reference publication should be defined through further annotation (not shown in the figure).

2.1 A graph-based approach to storing computational biology models

We suggest here to store SBML-encoded models in a graph-based database. The entry point for each model is the document node containing the information about the SBML schema level and version and the upload time and date. The model is then linked to a document node by a directed relation `hasModel`. In return, from the model node a directed relation `belongsTo` is set. The model node also serves as an "anchor" for following entities, ensuring easy traversal upwards from any point of the stored models. In obeying the hierarchical SBML structure, all further nodes are connected upwards with a `belongsTo` relation to their corresponding parent node. The model node stores the model name and id. It is possible to attach to it annotations like creators or publications. Part of the representation of the Michaelis-Menten model from above in Neo4J is shown in Figure 2 (left). The figure also shows the compartment node (with id and name) and its relation to the model node (`hasCompartment`); the three species nodes (P, S, E); and their relation to the model node (`hasSpecies`); and finally the information on species location in compartments (`isContainedIn`,`contains`). In a next step the reaction node is added and connected to the model node. For each reaction, species roles are defined as modifiers,

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level1" level="1" version="1">
  <model name="Michaelis_Menten_kinetics">
    <listOfCompartments>
      <compartment name="compartment" volume="1"/>
    </listOfCompartments>
    <listOfSpecies>
      <species name="S" compartment="compartment" initialAmount="1" boundaryCondition="true"/>
      <species name="E" compartment="compartment" initialAmount="0.999"
boundaryCondition="false"/>
      <species name="P" compartment="compartment" initialAmount="0.001"
boundaryCondition="false"/>
      <species name="P" compartment="compartment" initialAmount="0" boundaryCondition="true"/>
    </listOfSpecies>
    <listOfReactions>
      <reaction name="R" reversible="false">
        <listOfReactants>
          <speciesReference species="S" stoichiometry="1"/>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="P" stoichiometry="1"/>
        </listOfProducts>
        <listOfModifiers>
          <speciesReference species="E" stoichiometry="1" />
        </listOfModifiers>
        [..]
      </reaction>
        [..]
    </listOfReactions>
  </model>
</sbml>
```

Fig. 1. SBML code snippet showing the encoding of the Michaelis-Menten kinetics3: Enzyme (E) and substrate (S) form a complex (ES) resulting in enzyme and product.

reactants, or products (e.g., `is`/`hasProduct`, `is`/`hasModifier`). The same species may take part in different reactions with different roles. The extraction of model entities is resumed analogously with encoded parameters, events and other SBML concepts. The model entities may be further characterized by external knowledge to describe the underlying biology. SBML uses RDF triples, where each subject is an entity of interest; each predicate qualifies the relation to the external piece of knowledge (e.g., `is`, `hasPart`); and each object is a URI pointing to that piece of knowledge. This relation between entity and URI is also mapped to the database. Existing URIs are connected to the entity, for new URIs a node is created and linked. The same procedure holds for model creators and publication references: The corresponding nodes are only created if not yet contained in the database. Taken together, the sum of extracted information provides a detailed representation of the models' network structure and all annotations.

2.2 Integration of Simulation Setups

Recent works on model management, discussions in meetings and repeated calls for model reuse showed the importance of thorough management systems for computational models of biological systems. However, SBML is a format to encode the model structure. It does not, and did not intend to, cover concepts for the simulation experimental setups associated with a model. To address this need, another XML-based standard format has more recently been, called the Simulation Experiment Description Markup Language (SED-ML, [Wa11a]). SED-ML files contain pointers to one or more models, together with information how to treat a model to see a desired result.

[3] adapted from `http://www.gepasi.org/HMM.xml`

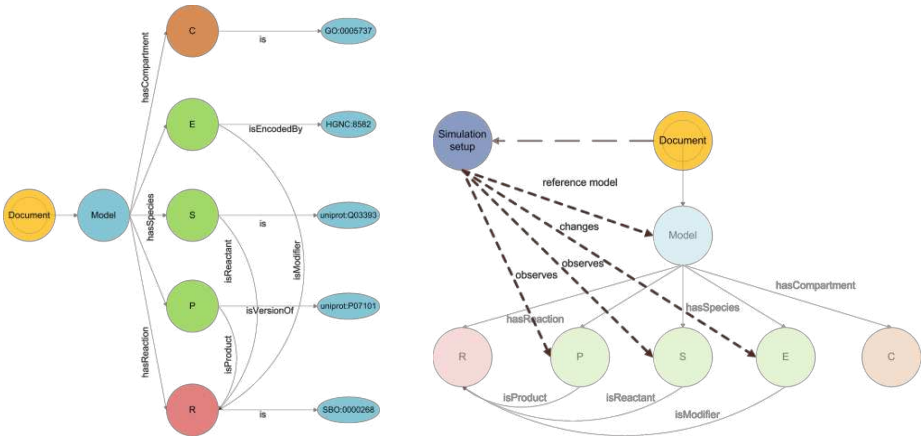

Fig. 2. Simplified view on a model representation in the Neo4J database. **Left:** The entry point is the Document node which is always related to one model. The model itself is then connected to different reactions (red), species (green) and compartments (brown) with specified relations (e.g., hasSpecies). For relations between species and reactions the roles `isReactant`, `isProduct` and `isModifier` are used. The links to ontology entries (blue) are also represented (e.g., `isVersionOf uniprot:P07101`). **Right:** Linking a SED-ML file to the model.

Efforts to encode the result data in standard format are ongoing, for example the Numerical Markup Language (NuML[4]). Considering the progress on simulation experiment encoding, we believe that a "model database" should not only encompass models but rather "models with associated simulation setups and result data". Only experimental setups together with a model make the model immediately reusable, and thereby enhance its usefulness. SED-ML files encode valuable information about links (URIs) to the models that the experiment runs on; changes applied to these models before simulation; the types of experiments that have been run; the simulation algorithm that has been used (through linking to the Kinetic Simulation Algorithm Ontology (KiSAO, [Co11]); and the output that was of interest and how it was generated. All the information needs to be stored in a way that it can be made explicitly available to the user; consequently it must be searchable, retrievable and viewable. To address the maintenance problem, we have recently launched a project called the Simulation Experiment Management System (SEMS[5]). It aims at developing an information management infrastructure for simulation experimental setups. As SED-ML is XML, we intend to reuse the approaches that we have developed for SBML model storage; in a first step, we integrate SED-ML files with the model data we already have in Neo4J. Most important in our opinion is the linkage of SED-ML files to model files, and later on result data: Explicit relations will be built in the graph database to link from a simulation task to the model used in that task (Figure 2, right), allowing us to link arbitrary numbers of models to arbitrary numbers of simulation setups. SED-ML furthermore stores all information on model updates prior to simulation, leading

[4] `http://code.google.com/p/numl/`

[5] `http://www.sbi.uni-rostock.de/sems`

to multiple descriptions applicable to one model. On the other hand, to compare results from several models, a SED-ML file may use more than one model to generate a particular output. A model, when published in a journal, should be able to reproduce all results shown in the figures of the article. More often than not, the model as, for example, available from BioModels Database, is capable of reproducing one particular figure with the structure provided in the model and with the initial conditions given. In order to reproduce the remaining figures, it is then necessary to change these conditions, e.g., to update concentrations of particular model entities. Relating simulation setups and models then allows linking n models and its entities to m experimental setups.

3 Discussion and Future Work

We have in this work discussed why a graph database is more suitable to represent the network structure in SBML models and associated simulation setups than is a relational approach. Compared to a relational storage approach (e.g., [Wa11b]) the graph-based approach does not demand the construction of many-to-many tables to establish the multiple relations between models constituents which then result in various join operations to re-construct a model from a retrieved model constituent. Contrarily, the concept of attributed nodes and relations allows for an easy and intuitive way to store models and associated meta-information. Another alternative to graph-based storage may be a document oriented storage approach, e.g. XML databases. However, we do not aim at keeping the exact hierarchical structure and format of the XML file, but rather focus on the biological network structure and its linkage to associated SED-ML files. A different approach is taken when transforming SBML and related formats into OWL [Li09] or RDF [Ho11]. Here, transformed models are stored in triple stores and accessed, for example, through SPARQL. Again, this approach does not allow easy integration of SED-ML files – a task that is particularly important for model storage.
In graph databases, even for a big number of models it is still possible to efficiently inter-relate models through common annotations, publications and persons. Finally, the graph database does not need to be adapted with every change in the underlying XML schemata, as would a relational database. We envision adding further model exchange formats other than SBML in the future. CellML, for example, has a similar structure that can be mapped to the database. The introduced storage approach will furthermore allow us to extend our current search approach to cover structure-based search in the future, which has not been possible in our recent ranked retrieval approach implemented in the demo search of BioModels Database [He10].
The integrated storage of models and their simulation setups for the first time enables search for suitable simulation experiments. As of now, no database provides a mechanism to store models together with their simulation setups. We have recently exemplified that the provision of simulation setups in SED-ML format is possible, for example as a meta-information to the models stored in BioModels Database. However, no system exists that associates the SED-ML file to more than one model file, thereby providing cross-linking to other models that might also have been used in the simula-

tion. Neither is there a system that provides the user with further information on the experiment. SED-ML files are not maintained inside a database. Consequently, they are not version controlled and there is no consistency check available. Mechanisms of informing users on changes of the underlying model are missing. We address all these issues in SEMS and will integrate our solutions with existing model databases in the forthcoming years.

4 Bibliography

[AG08] Angles R, Gutierrez C: Survey of graph database models. ACM Comput. Surv. 40 (2008)

[As00] Ashburner M, Ball C, Blake J, Botstein D, Butler H, others: Gene Ontology: tool for the unification of biology. Nature genetics 25:1 (2000)

[BS06] Bodenreider O, Stevens R: Bio-ontologies: current trends and future direc tions. Briefings in Bioinformatics 7:3 (2006)

[Co11] Courtot M, Nick J, Knüpfer C, Waltemath D, others: Controlled vocabularies and semantics in systems biology. Mol Syst Biol 7:543 (2011)

[Cu03] Cuellar A, Lloyd C, others: An overview of CellML 1.1, a biological model description language. Simulation 79:12 (2003)

[GGL03] Ghemawat S, Gobio H, Leung S: The Google File System; In: 19th ACM Symposium on Operating Systems Principles (2003).

[Gl10] Gleeson P, Crook S, others: NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail. PLoS computational biology 6:6 (2010)

[He10] Henkel R, Endler L, Peters A, Le Novère N, Waltemath D: Ranked retrieval of Computational Biology models. BMC Bioinf 11:423 (2010)

[Ho11] Hoehndorf R, Dumontier M, others: Integrating systems biology models and biomedical ontologies. BMC Systems Biology 5:1 (2011)

[Hu10] Hucka M, Bergmann F, Keating S, others: The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version (2010)

[Li09] Lister AL, Lord P, others: Annotation of SBML Models Through Rule-Based Semantic Integration. Journal of Biomedical Semantics 1 (2009)

[Li10] Li C, Donizelli M, others: BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. BMC systems biology 4:1 (2010)

[La98] Lassila O, Swick R, others: Resource description framework (RDF) model and syntax specification (1998)

[Sp11] Splendiani A, Rawlings CJ, Kuo S, Stevens R, Lord P: Lost in translation: data integration tools meet the Semantic Web, arxiv.org (2011)

[Vi10] Vicknair C, others: A comparison of a graph database and a relational data base: a data provenance perspective. Proceedings of the 48th Annual Souteast Regional Conference 42 (2010)

[Wa11a] Waltemath D, Adams R, Bergmann FT, others: Reproducible computational biology experiments with SED-ML. BMC Systems Biology 5:198 (2011)

[Wa11b] Waltemath D, Henkel R, others: Das Sombi-Framework zum Ermitteln geeigneter Suchfunktionen für biologische Modelldatenbasen. Datenbank-Spektrum 11:1 (2011)