# Combining computational models, semantic annotations, and associated simulation experiments in a graph database

Ron Henkel[1], Olaf Wolkenhauer[1,2],Dagmar Waltemath[1]

April 29, 2014

1. Department of Systems Biology and Bioinformatics, University of Rostock, Germany

2. Stellenbosch Institute for Advanced Study (STIAS), Wallenberg Research Centre at Stellenbosch University, Stellenbosch 7600, South Africa

* Corresponding authors: Ron Henkel, Dagmar Waltemath,
{ron.henkel|dagmar.waltemath}@uni-rostock.de

Subject categories: Computational Biology

Keywords: model repositories / graph database / bio-ontologies / SBML / CellML / SED-ML

Running Title: Integrative storage of models and associated data

# Abstract

Model repositories such as the BioModels Database or the CellML Model Repository are frequently accessed to retrieve computational models describing biological systems. However, the current designs of these databases limit the types of supported queries, and many data in these repositories cannot easily be accessed. Computational methods for model retrieval cannot be applied. In this paper we present a storage concept that meets this challenge. It grounds on a graph database, reflects the models' structure, incorporates semantic annotations and experiment descriptions, and ultimately connects different types of model-related data. The connections between heterogeneous model-related data and bio-ontologies enable efficient search via biological facts and grant access to new model features such as network structure. The introduced concept notably improves the access of computational models and associated simulations in a model repository. This has positive effects on tasks such as model search, retrieval, ranking, matching, filtering etc. We exemplify how CellML- and SBML-encoded models can be maintained in one database, how these models can be linked via annotations, and queried.






PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

# Introduction

Model repositories such as the BioModels Database (Li *et al*, 2010) and the CellML model repository (Yu *et al*, 2011) offer to the community valuable, curated, and resuable models describing biological systems. They enable researchers to study biological systems in the computer without necessarily implementing the models from scratch, thereby saving time, effort and money. In addition, curation has a positive effect on the quality of models used in modeling projects, because errors in the model encoding are more likely to be detected, they can be resolved and documented. Finally, model repositories use standard formats, e. g. the Systems Biology Markup Language (SBML) (Hucka *et al*, 2010) or CellML (Lloyd *et al*, 2004), to distribute ready-for-reuse models that can immediately be loaded in a large number of computational tools for simulation, analysis, visualization, or comparison (Hucka *et al*, 2011).

Each model describes certain aspects of a system. These aspects may be of functional, behavioral or structural nature (Knüpfer *et al*, 2013) and need to be covered in the description of the model. For example, the different models of the cell cycle in the BioModels Database contain biological entities and reactions that together characterize the cell division cycle. Semantic annotations relate model entities to external resources describing the underlying biology. A model of the cell cycle may be annotated with a term from Gene Ontology (GO) (Botstein *et al*, 2000) that defines the cell cycle biologically, e. g.,

> "The progression of biochemical and morphological phases and events that occur in a cell during successive cell replication or nuclear replication events. Canonically, the cell cycle comprises the replication and segregation of genetic material followed by the division of the cell, but in endocycles or syncytial cells nuclear replication or nuclear division may not be followed by cell division." (Gene Ontology, GO:0007049)

The SBML representation of this model is only equipped with the GO identifier (here: GO:0007049). This identifier, however, can be resolved computationally to access the full information from the Gene Ontology, making a semantic-based comparison of models feasible.

In the past years, the focus shifted from exchanging pure model code towards exchanging models and model-related information. As a consequence, we understand better what a model is about, the rationale behind building it, and ultimately how to reuse it. The necessary information to reuse a model is defined in the Minimum Information guideline for the annotation of models, MIRIAM (Le Novère *et al*, 2005). MIRIAM requires each biological entity in a model to be defined; links to the publication describing a model (denoted as reference publication); and instructions on how to use a model to reproduce a published result. In the following, we refer to these and other model-related information as meta-data.

Before the era of semantic knowledge integration and ontologies, model code contained only few meta-data. Thus models could be kept in file systems and relational data tables. This approach is unsuitable for today's models, because their meta-data is heterogeneous in structure and content. Consequently, it is difficult to map the meta-data onto relational tables with homogenous and pre-defined properties. This (technical) limitation results in many types of model-related data which are not extracted from the model. The information they contain is not accessible and thus lost for computational processing, e. g., when determining model similarities. Examples for neglected meta-data include the structure of the model (Henkel *et al*, 2012), model versions (Waltemath *et al*, 2013a), and associated simulation setups (Waltemath *et al*, 2013b). Given the efforts made to encode the biological knowledge in bio-ontologies, and then to link model entities with semantic information, the current situation is indeed unsatisfying. Often, one consequence of inaccessible meta-data is that the modeling results are not reproducible, because the link from model code to simulation experiment is missing (Waltemath *et al*, 2011b).

In this manuscript we propose the concept of graph databases for model storage and retrieval. Graph databases support heterogenuous data structures. They furthermore enable a flexible integration of model-related meta-data. We have focused our studies on models in SBML and CellML formats, associated simulation setups in SED-ML format, and semantic annotations from bio-ontologies. A key feature of our work is the *explicit linking*, which allows researchers to postulate queries across different data formats. The integration of model-related data, simulation experiments, semantic annotations and structure information supports modelers and biologists in finding models and reproducing scientific findings that are relevant to their own work. It fosters the exploration of published models and increases model reuse.



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

# Results

Model reuse can be improved if models *and* associated data are considered together. In this paper we present a novel storage concept that tightly links model code with model-related data. We focus on the data requested by two Minimum Information Guidelines: MIRIAM for requested information about models and MIASE , the Minimum Information About a Simulation Experiment (Waltemath *et al*, 2011a), for requested information about simulation setups. We store and link all data in a graph-database, where nodes contain the data, and edges represent the links between the data. The explicit linking of model-related data paves the way for new types of queries about models, e. g., "Return experiments observing entities representing a "'m-phase inducer phosphatase"' and acting as modifier in a reaction". This query runs on the different types of data that need to be linked to get the full picture of a model: model code (identifying all models that contain entity X); semantic annotations (identifying all entities X that are annotated with a term of a bio-ontology that is semantically similar to "m-phase inducer phosphatase"); the model's network structure (filtering those models where X is a modifier in a reaction); and finally associated simulation experiments (returning for each relevant model the possible simulations defined for it).

## Considered types of data and standard formats

Several types of data are relevant for a meaningful description of computational models in biology (Chelliah *et al*, 2009; Knüpfer *et al*, 2013; Waltemath *et al*, 2013b). Specifically, Knüpfer *et al* (2013) distinguish data for the extrinsic and intrinsic description of model function, behavior and structure. Many of these aspects have already been described in standard formats, including model structure, simulation descriptions, simulation results and semantic annotations. The development of standards is a continuous process, and their uptake by software tools and users progresses at different pace. For example, while many journals today recommend, or require, the provision of model code during submission (e. g., in SBML), there is no such recommendation to submit also a graphical representation in the Systems Biology Graphical Notations (SBGN) (Le Novère *et al*, 2009), nor to submit the simulation description (in SED-ML). Some formats are specified, but so far only used by a small number of software tools, e. g. the Systems Biology Result Markup Language (SBRML) (Dada *et al*, 2010). However, repeated calls for reproducibility of modeling results have been published in the past years. The development of standards fosters both, the submission of model-related data to model repositories such as BioModels Database and the distribution of archives such as the Research Objects Hettne *et al* (2013) or the recently launched COMBINE Archive (Waltemath *et al*, 2013b). In the following, we will only consider types of data that have been formally specified and for which curated data is available. These are basically the model code, simulation descriptions, semantic annotations and cross-references, and the mathematical characterization of models (Waltemath *et al*, 2013b).

**Model code in public model repositories:** Modelers use predominantly native programming languages, most commonly C or C++; script languages such as MATLAB or Python; and markup languages (XML) to describe their models. Program code and scripts, in general, are hard to understand and share. An XML representation reduces the obstacles to sharing data among diverse applications by providing a common format for expressing data structure and content (Seligman and Roenthal, 2001). XML formats for the standardized representation of models are SBML, CellML or NeuroML (Gleeson *et al*, 2010). They all focus on the encoding of the models' structure, for example the interactions in a pathway, and describe sets of entities and the processes between them. Hucka *et al* (2003) highlight the advantages of markup languages, in this case SBML, for model representations: Model definition becomes straight forward, and a tool chain is available.

Once in standard format, models can be published and shared through open repositories, and they can be easily used with a variety of simulation tools. BioModels Database guarantees persistence and long-term availability of ready-to-run models. To date it contains 490 curated models[1] and several thousands of automatically generated pathway models which have been generated from the KEGG database (Büchel *et al*, 2013). Many simulation tools read and write models in SBML (Hucka *et al*, 2011).

[1] release 26 of BioModels Database as of November 4th 2013



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

**Simulation setups:** While models are commonly shared through repositories, simulation experiments are not yet part of the standardization workflow (Cooper *et al*, 2014). However, the ability to represent increasingly complex biological phenomena requires models to be instantiated using different conditions, and these conditions must be formally described together with the model itself. For example, in pharmacometrics, the calculation of a parametrization of an individualized model is itself a complex procedure that requires the development of further standards (Swat *et al*, 2013).

To ensure the reproducibility of simulation results, the Simulation Experiment Description Markup Language (SED-ML) (Waltemath *et al*, 2011b) is an XML-based format that encodes the necessary information to reproduce a particular result. SED-ML Level 1 Version 1 (Waltemath *et al*, 2011c) enables the reproduction of time course simulations. After the recent update to Level 1 Version 2, SED-ML now covers more types of experiments, including pulse experiments and parameter scans (Bergmann *et al*, 2013). For selected models, both BioModels Database and the CellML Model Repository recently started to provide SED-ML files. Surprisingly though, these SED-ML files are not linked with the models inside the database. Therefore, the information about applicable simulation experiments, for example, cannot be derived for model retrieval, comparison or similar tasks that potentially improve model reuse. This is, however, desired by researchers who wish to define generic experimental setups, so-called virtual experiments (Cooper *et al*, 2014), and link these to sets of models for comparison, validation and functional curation Cooper *et al* (2011). Hence these types of meta-data are relevant for future database design decisions.

**Semantic annotations and cross-references:** Semantic annotations link model entities to terms in bio-ontologies. Ontologies, in general, are defined as specifications of a conceptualization (Gruber *et al*, 1993). Bio-ontologies, e. g. Gene Ontology, then are ontologies with a focus on biological terms. Many cross-references between ontologies are provided in BioPortal (Noy *et al*, 2009). Models in SBML and CellML use bio-ontologies to encode semantic annotations as RDF triples (Lassila *et al*, 1998). For example, an RDF triple could be added to the SBML species "X" and link it to the ontology term "ATP" in the ChEBI database for chemical interactions (Degtyarenko *et al*, 2008) (ID "CHEBI:15422"). So-called qualifiers specify the relation between entity and ontology term (Juty *et al*, 2012). A model entity could *be* ATP or *have a part* ATP. The sum of semantic annotations in a model describes its biological and mathematical background. In BioModels Database, models carry between three and 800 annotations, but on average 71 annotations, per model (Alm *et al*, 2014).

## A graph database for simulation models and associated data

If models encode networks - why do not we store them as graphs rather than using a relational approach? We found graph databases to be the best suitable concept to store models, because: *(1)* Many models in public databases encode networks that can be represented as graphs. *(2)* No unified schema exists for models and meta-data, making it difficult to define a relational database schema. *(3)* The highly linked models, entities within models, and meta-data are difficult to represent in a table-like relational database management system such as MySQL.

Traditionally, relational databases were developed for homogeneous, structured data, e. g. numerical data sets. Models, however, take various size and structure. SBML models in BioModels Database, for example, import data structures from external standards and link to entries in bio-ontologies. Among the external standards are *vCard*, electronic businesscards that identify the model author and curators (http://www.w3.org/TR/vcard-rdf/), or *Dublin Core*, a vocabulary mainly used to describe web resources (http://dublincore.org/). Some models are associated with simulation descriptions or graphical representations. Finding a relational representation of all these links and at the same time building an efficient database is not possible for such heterogeneous structures. Furthermore, relational databases are not designed to store semi-structured documents efficiently (Robinson *et al*, 2013). A core concept of relational databases is a fixed schema which defines the structure of the data they contain. Semi-structured documents (Buneman, 1997), however, have only loose constraints on the structure of the data. All XML formats are semi-structured, and so are SBML, CellML and SED-ML. Architectural choices in current model repositories date back to times when only a limited number of alternatives existed, standardization of external knowledge only began, and model files were only scarcely associated with meta-data. Since then the databases have



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

grown and functionality has been extended. The focus has shifted from model code to "model-related data". Interestingly, only a few systems' architectures have been revised.

NoSQL approaches, together with semantic web applications, more recently gained popularity in the Life Sciences (Splendiani *et al*, 2011), e. g., as Key-Value Stores, BigTable (Ghemawat *et al*, 2003), document databases, triple stores, or graph databases (Angles and Gutierrez, 2008). We will here focus on the graph database Neo4J (Vicknair *et al*, 2010). It is based on the concept of describing data in terms of nodes, edges and attributes. Nodes are connected via directed edges (relations) of certain types. Both, nodes and edges can then hold attributes. The Neo4J architecture follows the fundamental properties of databases, i.e. the ACID principles (atomicity, consistency, isolation, durability).

For demonstration purposes, we use here one of the early models in the BioModels Database, namely Tyson's model on cell division (Tyson, 1991). This model is fairly small, exists as SBML and CellML representation, and it is available from both BioModels Database[2] and the CellML Model Repository[3].

## Database design and data import

All data needs to be transformed into a representation of nodes and edges during import. The entry point for each data item in our database is a root node, which we call document root node. Attached to this node may either be one model (e. g. an SBML node), or a data item that is related to the model (e. g. a SED-ML node). The entry point for each ontology in our database is an ontology root node.

More specifically, **SBML models** are represented by a model node which serves as the anchor for all related model entities (Figure 1, left part). The model node stores the model's name (cyan in Figure 1) and id. Attached to the model node are annotation nodes, including the reference publication (purple and grey). The model node is also connected to reaction, species and compartment nodes to reflect the underlying structures in the biological network. The example in Figure 1 shows a subset of nodes and edges for the Tyson model. All information about these nodes is directly extracted from the model's SBML representation. The figure displays three species nodes (in green), one reaction node (in red) and one compartment node (in orange). The edge between the species node `pM` (a complex of phosphorylated Cyclin and phosphorylated cdc2) and the compartment node `Cell` represents the fact that the species `pM` is located in the compartment `Cell`. Because we have qualified relations in the SBML model, we can even be more specific and store the information that `pM` is linked to `Cell` via the relation `isContainedIn`. Further model entities are stored analogously, i.e. encoded parameters, events and other SBML concepts. Finally, the semantic annotations are extracted from the SBML model and stored. The use of graph databases makes this mapping intuitive: Nodes representing some model entity are linked to nodes representing a particular term in a bio-ontology. The edge specifies that relation. An additional node is created and connected each time a new URI is detected during model import. For our example, the species node `pM` is related via `hasPart` to the InterPro term `Diphthine synthase` (in grey). Taken together, the sum of extracted information provides a detailed representation of the models' network structure and all annotations.

**CellML models** represent networks of connections between so-called components. A component contains variables and mathematical relationships that manipulate those variables (Cuellar *et al*, 2003). This is a different, more abstract approach to representing reaction networks, and one of the reasons why an integrated storage of SBML and CellML models on the XML level is so difficult (Köhn and Strömbäck, 2008). Examples for CellML components are physical compartments, events, species, or other convenient modeling abstractions. As for SBML models, the entry point is a document node that is connected to a model node and serves as an anchor for the component nodes. Each model entity can be related to a semantic annotation. Figure 1 (right side) shows the representation of the CellML encoding of the Tyson model. Attached to the model node are the component nodes, for example `C2`, `Cp`, or `environment`. Each component holds a number of variables. These variables are mapped to corresponding variables of connected components, e.g. the variable `time` in component node `C2` is connected to the variable `time` in the `environment` node. Please note here that the model node links to the identical publication node as the SBML model. If existing, annotations are extracted from the CellML model and mapped to the database using the same URI scheme as with SBML models. While CellML models today are only sparsely annotated, several projects work towards

[2] `http://www.ebi.ac.uk/biomodels-main/BIOMD0000000005`

[3] `http://models.cellml.org/exposure/9bff394be3ade829feed94151b3d68b3/tyson_1991.cellml/view`



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

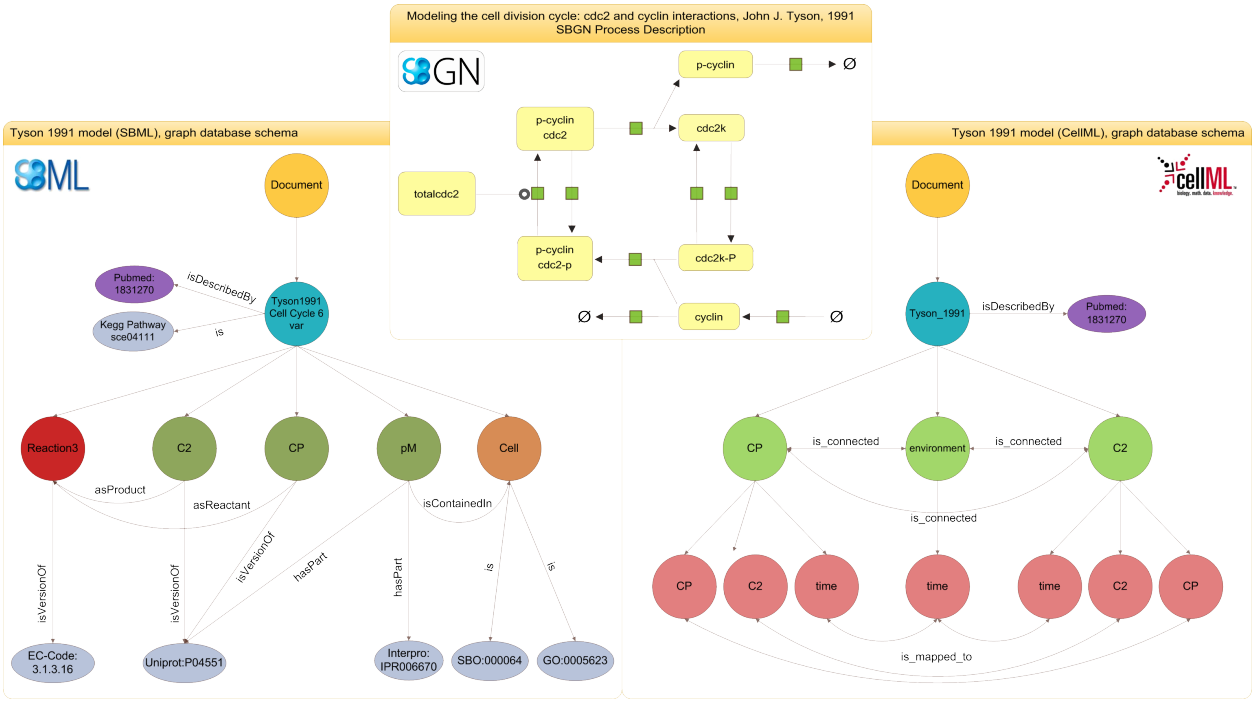


Figure 1: **Representations of the Tyson 1991 model.** The SBGN (top) representation shows the process description for the Tyson 1991 model. The graph database representation for the SBML encoded model is shown left, for CellML right. The document node is colored in yellow, model nodes in blue, annotation nodes in silver, and publication nodes in purple. For the SBML representation, reaction nodes are red, species nodes are dark green and compartment nodes are brown. For the CellML representation, component nodes are light green and variables are light red.

fully annotated CellML models (Wimalaratne *et al*, 2009; Gennari *et al*, 2011; de Bono *et al*, 2011). Our database is updated accordingly.

**SED-ML descriptions** specify simulation setups for models. They thereby link models, simulation algorithms, and output definitions (plots). A SED-ML description also explicitly declares the observed variables. In our design, the `SEDML` node serves as an anchor for one experiment. The `Modelreference` node links the experiment to all `Model` nodes used in the simulation. Figure 2 exemplifies how a model reference links one SED-ML description to an SBML and a CellML model. The algorithm used for the simulation is described by a term from the Kinetic Simulation Algorithm Ontology (KiSAO) (Courtot *et al*, 2011). KiSAO terms are imported into the database as terms from any other bio-ontology. A subset of KiSAO terms is depicted in Figure 2.

As aforementioned, **bio-ontologies** are integrated into the graph database to cover the semantic annotations in model representations and simulation descriptions. For example, model entities are annotated with domain knowledge from GO, ChEBI, UniProt; simulation descriptions contain links to simulation algorithms in KiSAO. Most bio-ontologies are available in the Web Ontology Language (OWL), which is a standard format for the representation of semantic information on the web. We parse these ontologies and add all concepts and relations as nodes and edges, respectively. Cross references are currently not mapped to the database, because these links cannot easily be determined in a reliable and consistent manner.

Table 1 summarizes the data types in our database and shows the number and size of the documents. Integration of further data resources is possible. For example, we provide an importer for ontologies encoded in OWL. However, the post-processing to link ontologies with models and simulation experiments needs to be done manually. Adding data encoded in SBRML (Dada *et al*, 2010) or NuML[4] would require additional importers and again a manual post-processing.

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



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

| Data domain | Documents | Nodes | Ontology | Nodes | Domain references |
|---|---|---|---|---|---|
| SBML | 462 | 91488 | KiSAO | 261 | 38 |
| CellML | 841 | 143521 | SBO | 606 | 8839 |
| SED-ML | 38 | 3352 | GO | 39787 | 7555 |

Table 1: Left: Number of files and stored nodes for each data domain. Right: Number of nodes for each stored ontology. The domain references state the number of links from a concept of an Ontology into a data domain. Here, all KiSAO concepts are linked to the domain of SED-ML while all SBO and GO concepts are referred to from the CellML or SBML domain.

### Linking model-related data

The main advantage of the graph-based concept described in the previous section is the possibility to define flexible links between the data domains. In concordance with previous considerations (Henkel *et al*, 2012; Waltemath *et al*, 2013b), we incorporate the following types of links:

(a) links between annotations (in SBML, CellML and SED-ML) and ontology entries,

(b) links between models (in SBML or CellML format) and SED-ML,

(c) links between model entities and SED-ML variables, and

(d) links between model entities from different model representation formats.

Figure 2 shows all existing links for the Tyson model. The database provides two encodings of the model, one in SBML and one in CellML format. Both representations are outlined on the left hand side of the figure. The first type of link is between model entities and ontology concepts (a). Here we only consider existing annotations. For each annotation in a model we add an explicit link to the data entry in the referenced bio-ontology. For example, based on the SBO annotation in the SBML model we build an additional edge between the node representing that annotation in SBML and the entry in SBO itself. Each concept (from an ontology) is only stored once but can be referred to by multiple model entities.

Another type of link is that between a model and a simulation description (b). When importing a SED-ML file into the database, we resolve the model references and check if those models are contained in our database. If this is the case, then additional edges are introduced for each model reference, between one model node and one SED-ML Modelreference node at a time. In the example in Figure 2, the original SED-ML file contained two model references, pointing to the Tyson 1991 model in SBML and CellML format, respectively. Thus we introduced two new edges in our database.

Furthermore, the variables of a DataGenerator in a SED-ML file may point to a specific entity in the referenced model. This pointer is used to identify the entity under observation, or for pre-processing before simulation. While we do not store the specific processing of a model entity, we keep the information if a model entity is part of a simulation. Consequently, a third type of link in our database is between the SED-ML Datagenerator node and a model entity (c). At the time of import, the SED-ML file is analyzed and the referenced species of the corresponding model is explicitly linked. Also, we flag species that are altered during SED-ML pre-processing (e. g., if the concentration of a species is changed). The links (a) – (c) can be established with information given in the documents. We regard them explicit links.

In addition, we determine implicit links between models of different representation formats (d). As we showed earlier, two models may link to the same publication (`Pubmed:1831270` in Figure 2). Our system concludes that, if two models share a publication, the entities of that model which are equally named share the same role in both model files. Even if the names are not fully identical but highly similar (e.g. in terms of Levenshtein Distance or stemming) it can be assumed that the entities are, in fact, identical. The confidence can be increased further if also the annotations match. For each such pair of entities, we add an additional edge to the database. Figure 2 shows the explicit connection of the entities `C2` in the SBML and `C2` in the CellML model. Both entities are linked because they have the same name, and they stem from the same reference publication.



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

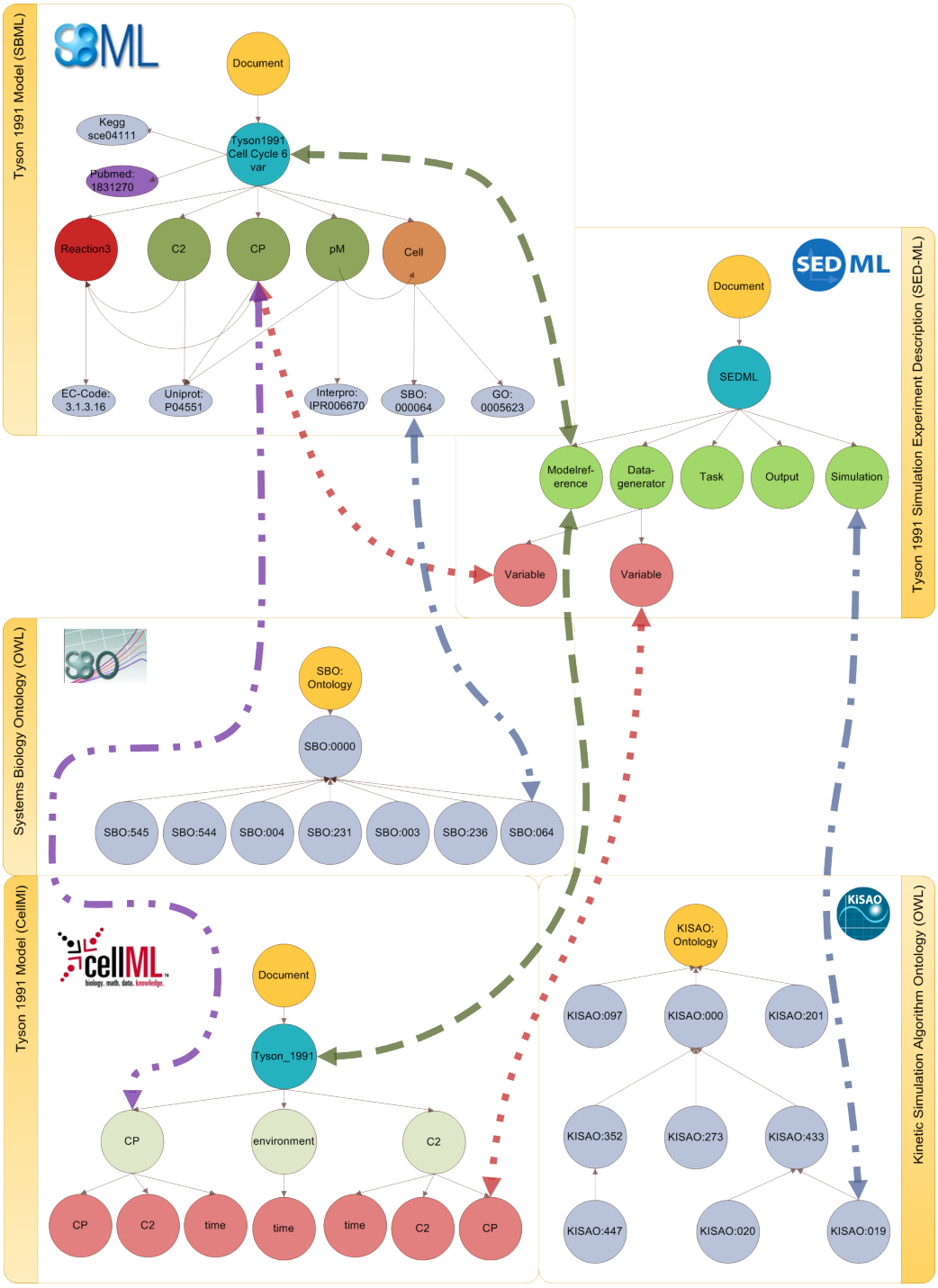


Figure 2: **Linking models, simulation descriptions and ontologies.** Linking between different data domains: simulation experiment descriptions and models (dashed line); defined observation variables and model entities (dotted line); annotated model entities and simulation experiment descriptions (dashed-dotted line); and model entities of different representation formats (double dotted-dashed line). The SBO example is explained in detail in the Implementation Section. The references to the simulation algorithm within a simulation experiment description are mapped to the corresponding entity in KiSAO.



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

# Discussion

## Advantages of implementing a graph-based concept

The main advantages of a graph-based concept for model storage are easy integration of heterogeneous resources; extensibility with further data resources; and improved model search.

**Realization of explicit linking between heterogeneous resources.** Currently, models and model-related data are only sparesly linked in the, predominantly relational, model repositories. Relational databases store data in tables and use the concept of primary and foreign keys to relate tables. Historically, they were designed for structured, homogenous data. However, they do not perform well on highly connected, semi-structured and heterogenous XML data. In a graph database, the integration of heterogeneous resources is straight forward. The concept of edges allows arbitrary connections to be defined by the creators of the database at any time. Particularly helpful for later model comparison are edges that connect nodes across model representation formats. For example, our database contains two representations of Tyson's 1991 cell cycle model, in SBML and in CellML, respectively. This link now becomes exploitable, because both model nodes share one publication node (PubMed:1831270). It is also useful to represent relations between a model and a simulation setup. Storing this information in the graph database allows modelers to quickly retrieve all models associated with a simulation experiment, and *vice versa*. For example, our graph database contains the information that there exists a SED-ML file which simulates and observes the change in concentration in `CP` in both encodings of the Tyson 1991 model (SBML and CellML) and then compares the simulation outcome (Figure 2). Finally, our database establishes links from model annotations into bio-ontologies. For example, the SBML model in Figure 2 contains the entity `Cell` which is annotated with a term from SBO. We can thus easily retrieve all models that are annotated with a particular ontology term. This is, for example, helpful in the classification of models (Alm *et al*, 2014).

**Extensibility of database schema.** Our graph database is schema optional. New data resources can efficiently be integrated and the database thus easily be extended. Specifically, we plan to integrate links to result data (in NuML format) and to Wet Lab descriptions once these exist in standard format. Data in NuML format could be linked to model entities, for example, when storing different parametrizations of a particular model.

**Possibility to incorporate model structure.** As current repositories do not represent the structure of a model, they cannot answer questions such as "Which model in the database contains the species that modifies most reactions?". To identify a species as the modifier of a reaction, this information must exist in the database. Figure 1 shows how we keep the information on the model structure: For each reaction in the model we map all reactants, modifier and products. Now a user can search for models that fulfill the following two conditions: (1) the species should only serve as a modifier in any of the model's reactions, and (2) only the topmost species per model should be considered. Our graph database retrieves the model "Schaber2012 - Hog pathway in yeast" [5], because the species `Hog1PPActive` occurs in ten reactions and only acts as a modifier (Query 1).

```
MATCH  (species:SBML_SPECIES)-[isMod:IS_MODIFIER]->()
WHERE  NOT((species)-[:IS_REACTANT]->() OR (species)-[:IS_PRODUCT]->())
WITH   species, count(isMod) AS numOfMod ORDER BY numOfMod DESC LIMIT 1
    MATCH species-[:BELONGS_TO]->model
    WHERE (model:SBML_MODEL)
RETURN model.NAME AS Model, species.NAME as Species, numOfMod
```

Query 1: Return the model with the most species acting only as a modifier.
Result 1: The model "Schaber2012 - Hog pathway in yeast" having the species Hog1PPActive which is acting as a modifier in ten reactions.

Graph databases offer further exciting applications, including the structure-based comparison of models. Combinations of nodes and edges form sub-networks which can for the first time be compared to each other using graph matching techniques. Once specific algorithms to map sub-models and identify suitable interfaces for automatized model coupling are in place, it will be possible to integrate them with our ranked retrieval system (Henkel *et al*, 2010). The Methods section contains additional examples for novel queries.

[5] originally from BioModels Database, `http://www.ebi.ac.uk/biomodels-main/BIOMD0000000429`



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

## Exploiting links to associated virtual experiments

Another type of query that cannot, as of now, be answered by current model repositories is: "Which simulation experiments for SBML and CellML models study the change of concentration in 'm- phase inducer phosphatase'?". To answer this question, it is not sufficient to query the model only. The repository must also support the retrieval of simulation experiments. In few cases, SED-ML files are provided in open repositories, enabling users to reproduce one or more figures of the reference publication. However, it is as of now not possible to include the SED-ML file in queries. Consequently, one cannot ask questions about the relation of a SED-ML file and a model. For example, Novak's 1997 cell cycle model can be run with two different setups, either reproducing Figure 2a or Figure 2b of Novak and Tyson (1997). Our approach stores the links between simulation setups and models and thus knows which experiments are applicable to which models (Query 2). When SED-ML descriptions are defined as templates for virtual experiments, and thus are applicable to classes of models (Cooper *et al*, 2014), it is also interesting – and now indeed possible – to ask the question: "Which models use this particular experiment description?". Furthermore, the links between SED-ML elements and KiSAO allow us to define restrictions on the SED-ML files we want to consider in a search result, e. g., to retrieve only models that can actually be simulated with a given simulation algorithm. With our system, the query "Which CellML encoded models can be simulated using a Livermore Solver?" can be answered (Query 3). One can also imagine to restrict results to changes in concentrations of a certain parameter.

```
MATCH  (m:SBML_MODEL)-[:REFERENCES_SIMULATION_MODEL]-ref-[:BELONGS_TO*2]->(sed:DOCUMENT)
WHERE  m.NAME='Novak1997 - Cell Cycle'
RETURN m.NAME AS Model, m.ID as ModelID, ref.MODELSOURCE as ModelSource, sed.FILENAME as SEDMLFile
```

Query 2: Return all simulations that can be applied to the model "Novak1997 - Cell Cycle"
Result 2: The requested model can be run by two simulations, reproducing Figure 2a and 2b by Novak and Tyson (1997)

```
MATCH  (sed:DOCUMENT)<-[:BELONGS_TO*2]-(sim:SEDML_SIMULATION)-[:SIMULATES]->
       (ref:SEDML_MODELREFERENCE)-[:REFERENCES_SIMULATION_MODEL]->m
WHERE  (sim.SIMKISAO='KISAO:0000019') AND filter(lable in labels(m) where lable ='CELLML_MODEL')
RETURN  m.NAME, sed.FILENAME
```

Query 3: Return only CellML models that can be simulated using a Livermore Solver (KISAO:0000019).
Result 3: The CellML encoded "Tyson 1991" model and the corresponding SED-ML file.

```
START  res=node:annotationIndex('RESOURCETEXT:(m-phase inducer phosphatase)')
MATCH  res<-[rel:is]-(a:ANNOTATION)-->(s:SBML_SPECIES)<-[:OBSERVES]-o-[:BELONGS_TO*]->(doc:DOCUMENT)
WITH   doc,res,s,o
    MATCH ()<-[:IS_MODIFIER]-s-[:BELONGS_TO]->m
RETURN DISTINCT doc.FILENAME AS SEDML, collect(distinct m.NAME) AS Model,
       collect(distinct res.URI) AS Resource, collect(distinct s.NAME) AS Species, collect(distinct o.TARGET) AS Target
```

Query 4: Return simulation descriptions observing a particular species that plays the role of a modifier or reaction, respectively. The observed species should be annotated as "m-phase inducer phosphatase" using the qualifier `is`.
Result 4: The result is shown and explained in Figure 3.

Strikingly, it is also possible to derive information from the graph database by combining the different data sets. Query 4 shows such a complex example. It combines index and structure information and spans data sets of ontology, models and simulation experiments. It retrieves simulation experiment description and corresponding models where a species is marked for observation by the simulation description. Additionally, the observed species must be annotated with a resource that `is` related to the phrase "m-phase inducer phosphatase" and the species must play the role of a modifier. The result is shown in Figure 3. To our knowledge this is the first time a system can answer queries spanned over different data sets and combining them with an index look-up.

## Statistics

Another interesting application for our graph database is to generate statistics about the integrated data. For example, we are interested in the most frequently used annotations for models in BioModels Database (Query 5) As the SBO is completely integrated, we find that the term SBO:0000009 is the most frequently used term for annotation. We can also compute the number of annotations using SBO:0000009 or one of its 125 children (Query 6). Finally, the system can also derive statistical values. For example, the average



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

number of annotations per model, as well as the minimum, maximum, and the standard derivation, can be computed for the set of SBML and CellML models available from BioModels Database and the CellML Model Repository (Query 7).

```
MATCH   (r:RESOURCE)-[qualifier:BELONGS_TO]->()
WITH    r, count(qualifier) AS AnnotationCount ORDER BY AnnotationCount DESC LIMIT 3
RETURN r.URI as Annotation, AnnotationCount
```

Query 5: What are the top-most three annotations used?
Result 5: Top three annotations used are SBO:0000009 (1127 times), SBO:0000252 (509 times), GO:0043241 (484 times)

```
MATCH   ()-[rel]->(res:RESOURCE)-[:IS_ONTOLOGY_ENTRY]-c-[:isA*0..]->s
WHERE   s.id="SBO_0000009"
RETURN  count(rel)
```

Query 6: How many annotations point to the term SBO:0000009 or one of its children?
Result 6: 3373 annotations pointing to SBO:0000009 or one of its children, 1127 of them point directly to SBO:0000009.

```
MATCH   (m:SBML_MODEL)<-[:BELONGS_TO*1..2]-(a:ANNOTATION)<-[:BELONGS_TO]-(r:RESOURCE)
WITH    m as Model, count(r) AS NumberOfAnnotation
RETURN max(NumberOfAnnotation), min(NumberOfAnnotation), avg(NumberOfAnnotation), stdev(NumberOfAnnotation)
```

Query 7: What is the minimum, maximum and average number of annotations per model?
Result 7: A model has a maximum of 800, a minimum of three and an average of 71 annotations.

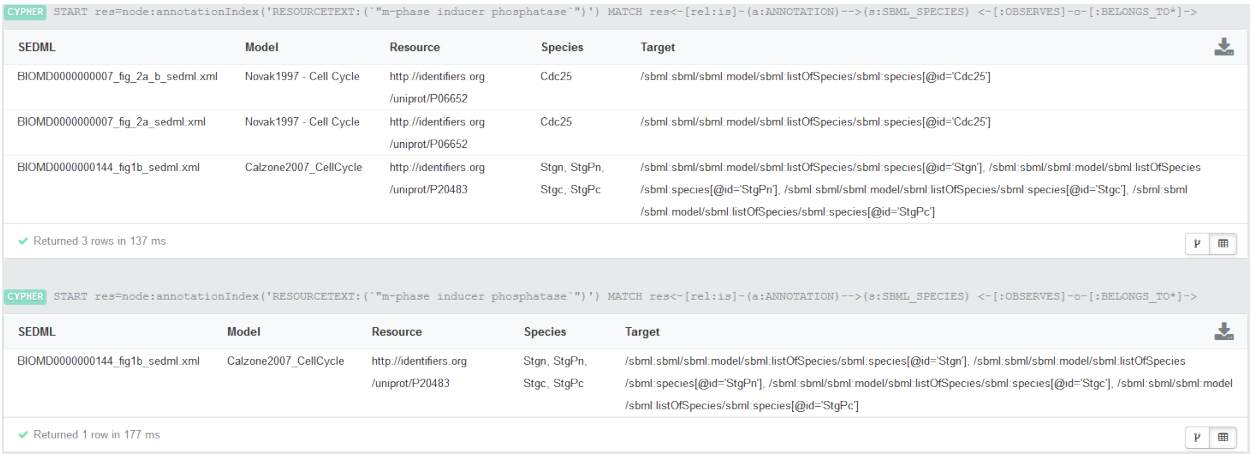


Figure 3: **Results for Query BM3.** The top query output shown in this Figure restricts the species role to *modifier*. Here, three SED-ML files are matching. The first and second belong to the same model and are observing the same species *Cdc25*. The third result is a SED-ML file where the observation of four species is encoded. The bottom query output shows the result of the same query but the species must be acting as a *reactant*. Here, only one SED-ML file is returned; the same as the third result of the top query output. Observed species' are annotated with a UniProt ID. *P06652* is the protein *Cdc25* in yeast whereas *P20483* is the protein *Stg* (*Cdc25*) in the fruit fly. Simulation files for CellML files are not retrieved, because CellML files are not yet fully annotated. If the CellML version of the Novak 1997 model had had annotations "m-phase inducer phosphatase", the database would have also returned the simulation description for that model.

## Comparison with other approaches

**RDF-triple-stores and SPARQL** Semantic Systems Biology has been termed a new field of research that aims to improve formal knowledge representation of computational models to enhance construction, comparison, validation, or retrieval (Dumontier *et al*, 2013). Several projects have started to convert model representations into semantically enriched formats to make the models comparable and to integrate the



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

knowledge about computational models better with knowledge contained in bio-ontologies. In general, the idea is to transform all data into RDF, store the RDF triples into several databases and provide SPARQL[6] endpoints to access the triples. For example, all models in BioModels Database have been converted into OWL representations before, using straight forward to more complex transformation methods (e. g., (Köhn and Strömbäck, 2008; Lister *et al*, 2010; Hoehndorf *et al*, 2011)). SPARQL has become the de-facto query language for the Semantic Web community and is also used in the domain of computational biology, e.g. Bio2RDF[7] (Belleau *et al*, 2008) or recently the BioModels Database SPARQL endpoint (Jupp *et al*, 2014a). While an in-depth comparison of graph-databases with RDF triple-stores (and associated query languages) is not in the scope of this paper, we could like to point out two disadvantages of RDF triple-stores and SPARQL in the domain of computational biology: First, in a triple-store everything is a triple of subject, predicate and object, e. g. `:a :isRelatedTo :b`. In consequence, it is not possible to distinguish links between entities (`:a :isVersionOf :P06652`) and simple entity properties (`:a :hasName :cdc25`). Second, triple-stores and SPARQL are tailored towards sub-graph retrieval. Thus, common graph algorithms like Dijkstra's algorithm (shortest path), directed path traversing, spanning trees or sophisticated graph matching patterns are hardly applicable on RDF triple-stores (Przyjaciel-Zablocki *et al*, 2012).

**BioModels Database** Recently, the European Bioinformatics Institute in Hinxton, UK, announced a SPARQL endpoint for BioModels Database[8]. All SBML encoded models in BioModels Database were converted into RDF representations and added to the EBI RDF Platform (Jupp *et al*, 2014b). To compare our concept against BioModels Database's approach to convert SBML models into RDF, we translated the query examples from the EBI web page into queries that we executed in our graph database. The results are shown in the following listings. Additionally, Figure 4 shows a visualization for Query BM3.

```
MATCH  (m:SBML_MODEL)-->(s:SBML_SPECIES)
WHERE  (m.ID="BIOMD0000000001")
RETURN m AS Model, collect(s.ID) as SpeciesID, collect(s.NAME) as SpeciesName
```

Query BM1: From model BIOMD0000000001, list all species identifiers and names
Result BM1: 12 species IDs (ALL, I, DL, ILL, D, DLL, B, BL, A, AL, IL, BL) and names (ActiveACh2, Intermediate, ...)

```
MATCH  (r:RESOURCE)-->()-[:BELONGS_TO]->(element)-->(m:SBML_MODEL)
WHERE  m.ID="BIOMD0000000001"
RETURN element.ID AS Element, LABELS(element) AS ElmentType, collect(r.URI) AS ElementAnnotation
```

Query BM2: Get element annotations of the model BIOMD0000000001
Result BM2: 104 annotations for 65 distinct elements, for example species ALL is annotated with IPR002394, GO:0005892 and SBO:0000297

```
MATCH  (r:RESOURCE)<-[rel]-()-->e-[:BELONGS_TO]->(m:SBML_MODEL)
WHERE  r.URI=~".*GO.*0005892"
RETURN m.ID AS ModelID, collect(e.ID) AS ElementIDs, type(rel) AS Qualifier, r.URI as URI
```

Query BM3: All model elements with annotations to acetylcholine-gated channel complex
Result BM3: From each model (BIOMD0000000001 and BIOMD0000000002) the same 12 species IDs are returned (ALL, I, DL, ILL, D, DLL, B, BL, A, AL, IL, BL), all are qualified with *isVersionOf*. A graphical representation is shown in Figure 4.

When comparing the query examples and results, we could easily reproduce all queries and results using our system. One of the original SPARQL queries corresponds to Query BM3, asking for models annotated with the "acetylcholine-gated channel complex". Due to the missing index support for this RDF store, the user must manually look-up and transform this annotation term into a URL, and paste that into the query. Our system, in the contrary, is able to retrieve this information automatically by a simple index-based query. A detailed example for the automatic conversion of an annotation is given in the aforementioned Query 4 or in Query M6 in the Methods section.

**COMBINE Archive** The types of meta-data considered in this work are also agreed upon by an effort called the COMBINE archive (Waltemath *et al*, 2013b), which aims at publishing extractable archive files that then contain all files necessary to reproduce a scientific modeling result in the life sciences. However,

[6] `http://www.w3.org/TR/rdf-sparql-query/`
[7] `http://bio2rdf.org/`
[8] `http://www.ebi.ac.uk/rdf/services/biomodels/`



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

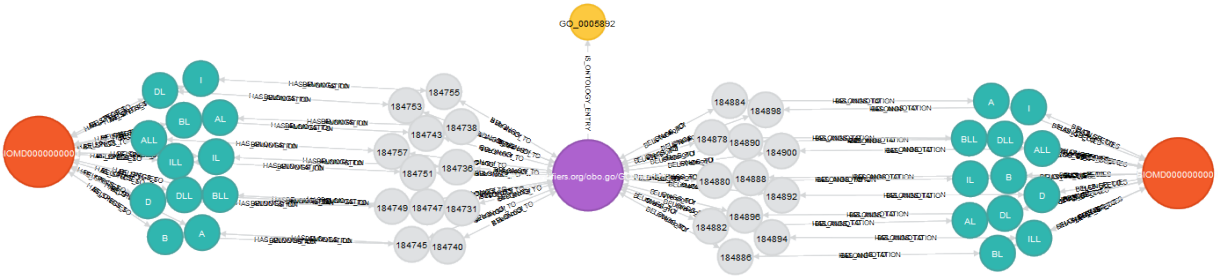

Figure 4: **Visualization for Query BM3.** The centered, purple node is the requested annotation GO:0005892, the green nodes are the species linked to the GO annotation, the orange nodes are the models "Edelstein1996 - EPSP ACh event" (BIOMD0000000001) and "Edelstein1996 - EPSP ACh species" (BIOMD0000000002).

we argue that the COMBINE archive is not a solution for the management of those files, but contributes to their export and exchange among researchers.

We conclude that our system incorporates and links knowledge that is in principle already available in public repositories, but not yet utilized. The knowledge is encoded in meta-data, in particular links to simulation experiments and semantic annotations with terms from bio-ontologies. The key to using this knowledge in model management tasks is its explicit linking and indexing in the database. We have demonstrated how relevant meta-data can be stored in a graph-database such as Neo4J, and we have exemplified how the meta-data can subsequently improve model retrieval and thus model reuse.

Our concept is easy to adapt and implement. An interface to test and query the database described in this paper is available.[9] In addition, a web API[10] designed to search SBML, CellML and SED-ML files is available for testing. A prototype implementation is running as a search service on an instance of the Physiome Model Repository, which is the backend of the CellML Model Repository.[11]

# Materials and Methods

## Mapping XML encoded models and model-related data to the graph database

When importing models to the database, the database entry point is a document node. This document node links to a model node via the directed edge `hasModel`. The model node has a model name and relations (i. e., edges) to nodes that represent model entities. In the case of SBML these entities include species, compartments, or reactions. For example, a model's species is represented by its own node. Additionally, an edge from the model node to the species node is created and named `hasSpecies`. Nodes for each reaction and compartment are created and connected with `hasReaction` and `hasComartment`, respectively. Moreover, relations of model elements are mapped to the graph database, i .e. a species node is connected to a compartment node with `isContainedIn`. To ensure an easy traversal upwards, a connection is created from each node of the stored model that points to the parent of the current node. The corresponding edges are named `belongsTo`. Furthermore, it is possible to attach an annotation to each model entity, describing the particular entity in more detail. All such annotations are stored to the database and indexed. The textual descriptions of terms in ontologies such as GO or ChEBI are retrieved from the according web pages, indexed and then processed. This index is afterwards used for ranked model retrieval as described in (Henkel *et al*, 2010). Attached to every node is a so-called label that names the type of node, e. g. species, compartment or annotation. Labels are indexed and allow to select all nodes of a specific type.

[9] `https://sems.uni-rostock.de/projects/masymos/`
[10] `https://sems.uni-rostock.de/projects/morre/`
[11] `http://staging.physiomeproject.org/`



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

## Implementation

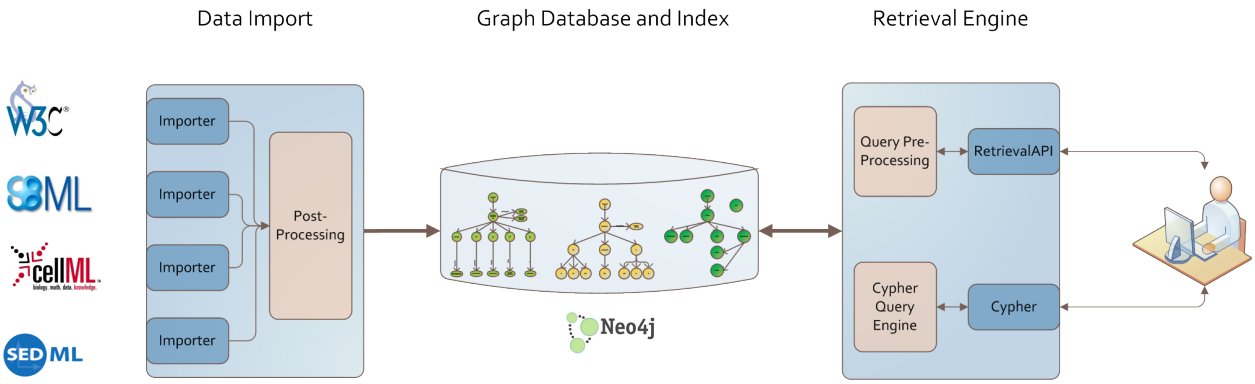


Figure 5: **Architecture of our graph database.** Data from different models, simulation descriptions or ontologies is imported using a format dependent importer. Each import undergoes a post processing afterwards. The stored graph and index structures are available via two RestAPI based retrieval interfaces, Cypher and one an adaption of (Henkel *et al*, 2010). The data itself is stored in a Neo4J graph database.

We have implemented the graph-based storage according to the architecture depicted in Figure 5. The Neo4J[12] database stores model files, simulation descriptions and model-related information in a graph manner. The retrieval engine is based on the ranked retrieval described in (Henkel *et al*, 2010). It allows users to access the data in the database, and retrieve ranked lists of results for their text queries. Queries are resolved using the Lucene framework[13], and ranked based on predefined similarity features. The data import pushes different data formats, including model code, simulation experiment descriptions and ontologies, into the graph database. Afterwards a post-process takes care of linking the added data of different domains.

**Models and Simulation Descriptions** are added to the database using format-dependent importers. Each supported format has its specification. Consequently, importers were implemented for SBML (based on jSBML (Dräger *et al*, 2011)), for CellML, and for SED-ML (based on jlibsedml (Waltemath *et al*, 2011b)). All importers share a common interface which maps the model and simulation files onto a graph structure. The import keeps the models' semantic information and all content that is relevant for later model querying, retrieval and display.

**Bio-ontologies** available in OWL can also be imported using the owl-api[14] and the JFact[15] reasoner. However, after adding an ontology to the database a post-processing is required to link model or simulation description entities to the newly added Ontology concepts. This post-processing is part of the linking process.

**Linking models and simulations** is done using the graph query and data modeling language Cypher[16] (Robinson *et al*, 2013), which is shipped along with Neo4J. The following query shows the command to link SBO annotations of models to the corresponding concepts of the SBO using Cypher.

```
MATCH  (res:RESOURCE), (sbo:SBOOntology)
WHERE  (res.URI =~ ".*SBO.*") AND (RIGHT(res.URI, 7) = RIGHT(sbo.id, 7))
CREATE res-[link:IS_ONTOLOGY_ENTRY]->sbo
RETURN count(link);
```

Query P1: Select and match and link the SBO annotations extracted from models with corresponding concepts from the SBO.
Result P1: The number of created links.

The *MATCH* clause selects every node that is labeled with the term "RESOURCE" and "SBOOntology" into the variable `res` and `sbo`, respectively. The *WHERE* clause restricts the selection to only those nodes satisfying the following constraints. In this case, the attribute URI of a resource node must contain the string

[12] `http://www.neo4j.org/`
[13] `http://lucene.apache.org/core/`
[14] `http://owlapi.sourceforge.net/`
[15] `http://jfact.sourceforge.net/`
[16] `http://www.neo4j.org/learn/cypher`



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

"SBO" and the last seven digits must correspond to the last seven digits of a node id out of the selected SBO concepts. This pairs all SBO annotations used in a stored model with the corresponding entry within the SB-Ontology. For the selected pairs of nodes the *CREATE* clause adds a new directed edge to the graph connecting both nodes. The label of the selected edge is `IS_ONTOLOGY_ENTRY`. Finally, the *RETURN* clause counts the number of edges created by this command and returns it to the user. A similar procedure applies to other bio-ontologies.

## Supported types of queries

The Cypher Query Language provides direct access to the data in our graph database. Cypher is the declarative language to pose queries against graph structures, similarly to SQL for relational databases. Our system supports standard queries such as data look-ups, filtering and aggregation. In addition, more complex queries regarding the model's structure can be posed.

**Look-up and filtering.** Database look-ups and filtering are shown in Query M1 and in Query M2. In these examples, the *MATCH* clause uses a build-in index to retrieve all nodes labeled as CellML model (Query M1) while the *WHERE* clause restricts the nodes to the ones matching the given name (Query M2). The result of the first query is a list of 841 stored CellML models, while the second query returns only the Tyson 1991 model.

```
MATCH  (m:CELLML_MODEL)
RETURN m
```

Query M1: Database look-up. Return all CellML models
Result M1: List of 841 models

```
MATCH  (m:CELLML_MODEL)
WHERE  m.NAME = 'tyson_1991'
RETURN m
```

Query M2: Database look-up and filtering. Return CellML models with the name '"tyson_1991"
Result M2: A model node containing the attribute NAME:"tyson_1991"

**Graph matching.** Query M3 shows how graph structures can be queried. In this example, all components from the Tyson 1991 model are selected. Eight component names are returned, as denoted in the *RETURN* clause.

```
MATCH  (m:CELLML_MODEL)-->(c:CELLMLCOMPONENT)
WHERE  m.NAME = 'tyson_1991'
RETURN c.NAME
```

Query M3: Database graph structure query. Select the aforementioned Tyson model and return all its components.
Result M3: The components YP, Y, M, pM, CP, C2, environment and reaction_constants.

**Aggregation.** In SQL, aggregation functions such as `count()` or `sum()` group values from multiple rows into a single value. Query M4, shows how to define a query that counts the number of species for each model in the graph database. The *MATCH* clause selects the Tyson 1991 model, all connected components and variables. The *RETURN* clause counts and returns the number of variables for this model. Further examples of aggregation queries have been shown in Table 1 in the Results section.

```
MATCH  (m:CELLML_MODEL)-->(c:CELLMLCOMPONENT)-->(v:CELLMLVARIABLE)
WHERE  m.NAME = 'tyson_1991'
RETURN count(v)
```

Query M4: Database aggregation query. Count the number of variables contained by any component of the aforementioned Tyson model
Result M4: This model has 68 variables.

**Statistics** We provide statistics as another type of queries. Query M5 returns the minimum, maximum and average for the number of variables attached to components in CellML models. To provide these statistic



PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014

values, elements (in this case the CellML components) are selected and bound to an aggregation value using the *WITH* clause.

```
MATCH  (m:CELLML_MODEL)-->(c:CELLMLCOMPONENT)-->(v:CELLMLVARIABLE)
WITH   c as component, count(v) as NumOfVar
RETURN min(NumOfVar), max(NumOfVar), avg(NumOfVar), stdev(NumOfVar)
```

Query M5: Statistics query. Retrieve minimum, maximum average and standard derivation of for the number of variables attached to a component.
Result M5: A minimum of one and a maximum of 431 variables are attached to a component of a CellML model. On average each component has 9.64 variables attached with a standard derivation of almost 16.

**Index support.** Finally, Query M6 uses an index to retrieve nodes matching a given pattern. The indexed annotations are queried for the term "m-phase inducer phosphatase" using the *START* clause.

```
START  res=node:annotationIndex('RESOURCETEXT:(m-phase inducer phosphatase)')
RETURN res
```

Query M6: Database index query. Retrieve all annotations containing the phrase "m-phase inducer phosphatase"
Result M6: A set of seven resources (InterPro IPR000751; Enzyme Commission number 3.1.3.48; and UniProt: P30311, P23748, P20483, P06652, P30304)

## Database scaling

Büchel *et al* (2013) describe how to build computational models from biochemical pathway maps. The path2models[17] project resulted in more than 140.000 SBML models of a total size of 70GB. We used this data-set to challenge the database's performance on an average office system (Intel Core 2 Quad @ 2.66 GHz CPU, 8 GB RAM, Windows 7 64 Bit). The database was created in 20 hours and 40 min, thus every model required 531 ms on average. While importing the path2models project, 45.5 million nodes and 492 million relationships where created; the database size is approximately 83GB, including the indices.

# Acknowledgements

The authors thank Rebekka Alm for implementing part of the SED-ML storage and Markus Wolfien for creating and encoding the SED-ML files used in this paper.

# References

Alm R, Waltemath D, Wolkenauer O, Henkel R (2014) Annotation-based feature extraction from sets of SBML models. In *Databases in the life sciences (DILS), accepted for publication*

Angles R, Gutierrez C (2008) Survey of graph database models. *ACM Computing Surveys CSUR* **40**: 1

Belleau F, Nolin MA, Tourigny N, Rigault P, Morissette J (2008) Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics* **41**: 706–716

Bergmann FT, Cooper J, Nickerson D, Le Novère N, Waltemath D (2013) Simulation Experiment Description Markup Language (SED-ML): Level 1 Version 2 (Specification)

Botstein D, Cherry J, Ashburner M, Ball C, Blake J, Butler H, Davis A, Dolinski K, Dwight S, Eppig J, *et al* (2000) Gene Ontology: tool for the unification of biology. *Nat Genet* **25**: 25–29

Büchel F, Rodriguez N, Swainston N, Wrzodek C, Czauderna T, Keller R, Mittag F, Schubert M, Glont M, Golebiewski M, *et al* (2013) Large-scale generation of computational models from biochemical pathway maps. *arXiv preprint arXiv13077005*

Buneman P (1997) Semistructured data. *Proceedings of the sixteenth ACM SIGACT SIGMOD SIGART symposium on Principles of database systems* : 117–121

[17] http://code.google.com/p/path2models/

514 Chelliah V, Endler L, Juty N, Laibe C, Li C, Rodriguez N, Le Novère N (2009) Data integration and semantic
515 enrichment of systems biology models and simulations. In *Data Integration in the Life Sciences*. Springer

516 Cooper J, Mirams GR, Niederer SA (2011) High-throughput functional curation of cellular electrophysiology
517 models. *Progress in biophysics and molecular biology* **107**: 11–20

518 Cooper J, Vik JO, Waltemath D (2014) A call for virtual experiments: accelerating the scientific process.
519 *PeerJ PrePrints urlhttpspeerjcompreprints273*

520 Courtot M, Juty N, Knüpfer C, Waltemath D, Zhukova A, Dräger A, Dumontier M, Finney A, Golebiewski
521 M, Hastings J, *et al* (2011) Controlled vocabularies and semantics in systems biology. *Molecular systems*
522 *biology* **7**

523 Cuellar A, Lloyd C, Nielsen P, Bullivant D, Nickerson D, Hunter P (2003) An Overview of CellML 1.1, a
524 Biological Model Description Language. *Simulation* **79**: 740–747

525 Dada JO, Spasić I, Paton NW, Mendes P (2010) SBRML: a markup language for associating systems biology
526 data with models. *Bioinformatics* **26**: 932–938

527 de Bono B, Hoehndorf R, Wimalaratne S, Gkoutos G, Grenon P (2011) The RICORDO approach to semantic
528 interoperability for biomedical data and models: strategy, standards and solutions. *BMC Research Notes*
529 **4**: 313

530 Degtyarenko K, de Matos P, Ennis M, Hastings J, Zbinden M, McNaught A, Alcántara R, Darsow M, Guedj
531 M, Ashburner M (2008) ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic*
532 *acids research* **36**: D344–D350

533 Dräger A, Rodriguez N, Dumousseau M, Dörr A, Wrzodek C, Le Novère N, Zell A, Hucka M (2011) JSBML:
534 a flexible Java library for working with SBML. *Bioinformatics* **27**: 2167–2168

535 Dumontier M, Chepelev LL, Hoehndorf R (2013) Semantic Systems Biology: Formal Knowledge Representa-
536 tion in Systems Biology for Model Construction, Retrieval, Validation and Discovery, In *Systems Biology*,
537 XXXI, Netherlands: Springer, pp. 355–373

538 Gennari JH, Neal ML, Galdzicki M, Cook DL (2011) Multiple ontologies in action: Composite annotations
539 for biosimulation models. *Journal of biomedical informatics* **44**: 146–154

540 Ghemawat S, Gobioff H, Leung ST (2003) The Google file system. *SIGOPS Oper Syst Rev* **37**: 29–43

541 Gleeson P, Crook S, Cannon RC, Hines ML, Billings GO, Farinella M, Morse TM, Davison AP, Ray S, Bhalla
542 US, *et al* (2010) NeuroML: a language for describing data driven models of neurons and networks with a
543 high degree of biological detail. *PLoS computational biology* **6**: e1000815

544 Gruber TR, *et al* (1993) A translation approach to portable ontology specifications. *Knowledge acquisition*
545 **5**: 199–220

546 Henkel R, Endler L, Peters A, Le Novère N, Waltemath D (2010) Ranked retrieval of Computational Biology
547 models. *BMC bioinformatics* **11**: 423

548 Henkel R, Le Novère N, Wolkenhauer O, Waltemath D (2012) Considerations of graph-based concepts to
549 manage of computational biology models and associated simulations. *Proceedings of the 2012 GI Jahresta-*
550 *gung* : 1545–1551

551 Hettne KM, Dharuri H, Zhao J, Wolstencroft K, Belhajjame K, Soiland-Reyes S, Mina E, Thompson M,
552 Cruickshank D, Verdes-Montenegro L, *et al* (2013) Structuring research methods and data with the Re-
553 search Object model: genomics workflows as a case study. *arXiv preprint arXiv13112789*

554 Hoehndorf R, Dumontier M, Gennari JH, Wimalaratne S, de Bono B, Cook DL, Gkoutos GV (2011) Inte-
555 grating systems biology models and biomedical ontologies. *BMC systems biology* **5**: 124

Hucka M, Bergmann FT, Keating SM, Schaff JC, Smith LP (2010) The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core

Hucka M, Bergmann FT, Keating SM, Smith LP (2011) A profile of today's SBML-compatible software. In *e-Science Workshops (eScienceW), 2011 IEEE Seventh International Conference on*. IEEE

Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, *et al* (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19**: 524–531

Jupp S, Malone J, Bolleman J, Brandizi M, Davies M, Garcia L, Gaulton A, Gehant S, Laibe C, Redaschi N, Wimalaratne SM, Martin M, Le Novère N, Parkinson H, Birney E, Jenkinson AM (2014a) The EBI RDF Platform: Linked Open Data for the Life Sciences. *Bioinformatics* **30**

Jupp S, Malone J, Bolleman J, Brandizi M, Davies M, Garcia L, Gaulton A, Gehant S, Laibe C, Redaschi N, *et al* (2014b) The EBI RDF platform: linked open data for the life sciences. *Bioinformatics* : btt765

Juty N, Le Novère N, Laibe C (2012) Identifiers. org and MIRIAM Registry: community resources to provide persistent identification. *Nucleic acids research* **40**: D580–D586

Knüpfer C, Beckstein C, Dittrich P, Le Novère N (2013) Structure, function, and behaviour of computational models in systems biology. *BMC systems biology* **7**: 43

Köhn D, Strömbäck L (2008) A Method for Semi-automatic Standard Integration in Systems Biology. In *Database and Expert Systems Applications*. Springer

Lassila O, Swick RR, *et al* (1998) Resource Description Framework (RDF) model and syntax specification

Le Novère N, Finney A, Hucka M, Bhalla US, Campagne F, Collado-Vides J, Crampin EJ, Halstead M, Klipp E, Mendes P, *et al* (2005) Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature biotechnology* **23**: 1509–1515

Le Novère N, Hucka M, Mi H, Moodie S, Schreiber F, Sorokin A, Demir E, Wegner K, Aladjem MI, Wimalaratne SM, *et al* (2009) The systems biology graphical notation. *Nature biotechnology* **27**: 735–741

Li C, Donizelli M, Rodriguez N, Dharuri H, Endler L, Chelliah V, Li L, He E, Henry A, Stefan MI, *et al* (2010) BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC systems biology* **4**: 92

Lister AL, Lord P, Pocock M, Wipat A (2010) Annotation of SBML models through rule-based semantic integration. *J Biomed Semantics* **1**: S3

Lloyd CM, Halstead MD, Nielsen PF (2004) CellML: its future, present and past. *Progress in biophysics and molecular biology* **85**: 433–450

Novak B, Tyson JJ (1997) Modeling the control of DNA replication in fission yeast. *Proceedings of the National Academy of Sciences* **94**: 9147–9152

Noy NF, Shah NH, Whetzel PL, Dai B, Dorf M, Griffith N, Jonquet C, Rubin DL, Storey MA, Chute CG, *et al* (2009) BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research* **37**: W170–W173

Przyjaciel-Zablocki M, Schätzle A, Hornung T, Lausen G (2012) Rdfpath: Path query processing on large rdf graphs with mapreduce. In *The Semantic Web: ESWC 2011 Workshops*. Springer

Robinson I, Webber J, Eifrem E (2013) *Graph Databases*. Edition 1. CA, USA: O'Reilly Media

Seligman L, Roenthal A (2001) XML's impact an databases and data sharing. *Computer* **34**: 59–67

Splendiani A, Rawlings CJ, Kuo SC, Stevens R, Lord PW (2011) Lost in translation: data integration tools meet the Semantic Web (experiences from the Ondex project). *CoRR* **abs/1103.4749**

Swat M, Moodie S, Kristensen NR, Le Novère N, *et al* (2013) PharmML–An Exchange Standard for Models in Pharmacometrics (Specification)

Tyson JJ (1991) Modeling the cell division cycle: cdc2 and cyclin interactions. *Proceedings of the National Academy of Sciences* **88**: 7328–7332

Vicknair C, Macias M, Zhao Z, Nan X, Chen Y, Wilkins D (2010) A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th annual Southeast regional conference*. ACM

Waltemath D, Adams R, Beard DA, Bergmann FT, Bhalla US, Britten R, Chelliah V, Cooper J, Crampin EJ, *et al* (2011a) Minimum information about a simulation experiment (MIASE). *PLoS computational biology* **7**: e1001122

Waltemath D, Adams R, Bergmann FT, Hucka M, Kolpakov F, Miller AK, Moraru II, Nickerson D, Sahle S, Snoep JL, *et al* (2011b) Reproducible computational biology experiments with SED-ML-the simulation experiment description markup language. *BMC systems biology* **5**: 198

Waltemath D, Bergmann FT, Adams R, Le Novère N (2011c) Simulation Experiment Description Markup Language (SED-ML): Level 1 Version 1 (Specification)

Waltemath D, Henkel R, Hälke R, Scharm M, Wolkenhauer O (2013a) Improving the reuse of computational models through version control. *Bioinformatics* **29**: 742–748

Waltemath D, Wolkenhauer O, Le Novère N, Dumontier M (2013b) Possibilities for Integrating Model-related Data in Computational Biology. In *Proceedings of the 9th International Conference on Data Integration in the Life Sciences*. CEUR Workshops

Wimalaratne SM, Halstead MD, Lloyd CM, Crampin EJ, Nielsen P (2009) Biophysical annotation and representation of CellML models. *Bioinformatics* **25**: 2263–2270

Yu T, Lloyd CM, Nickerson DP, Cooling MT, Miller AK, Garny A, Terkildsen JR, Lawson J, Britten RD, Hunter PJ, *et al* (2011) The physiome model repository 2. *Bioinformatics* **27**: 743–744





PeerJ PrePrints | http://dx.doi.org/10.7287/peerj.preprints.376v2 | CC-BY 4.0 Open Access | rec: 6 Oct 2014, publ: 6 Oct 2014