PhD Thesis

Contributions to the reuse and reproducibility of computational biology models

by

Dipl.-Inf. Ron Henkel

Dissertation zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik

Submitted by:	Dipl.-Inf. Ron Henkel
Date of Birth:	1982-09-19 in Waren (Müritz)
Examiners:	Prof. Dr. Kurt Sandkuhl, Universität Rostock, Institut für Informatik
	Prof. Dr. Jacky Snoep, Stellenbosch University, Department of Biochemistry
Einreichung:	Rostock, November 18th 2022
Verteidigung:	Rostock, June 14th 2023

https://doi.org/10.18453/rosdok_id00004411

Abstract

The ability to reproduce research is at the heart of science. Results can only be scientifically relevant if it can be reproduced, providing that experimental settings are coherently reported. This also holds true for computational research, such as models developed to gain deeper insights into biological systems as it is done in the life sciences and systems biology in particular.

With the rapidly increasing number of available models and new ones being developed, crucial tasks for any researcher in this field are to find models of relevance, to keep track of changes in a model of interest, and to ultimately run the model in order to reproduce scientific findings from in a publication. However, with the current storage and organisation of models, those tasks are in the best case tedious, thus rendering model reuse and reproducibility of modeling results nearly impossible.

This thesis contributes to improving reuse and reproducibility of computational biology models by suggesting and implementing innovative and efficient means for model storage, model retrieval and comparison, and model provenance. The model management concepts developed and implemented here, provide a graph-based model storage and a sophisticated model retrieval framework, two crucial steps towards a FAIR model management. In addition, as the storage and retrieval concept also include model meta-information from a variety of domains, the thereby created knowledge graph is able to provide results for domain-spanning queries. Having such a knowledge graph at hand offers a variety of new research possibilities, e.g., for model analysis and data exploration.

With a scientifically broader view and looking into the future, expanding and integrating concepts and implementations developed in context of this thesis towards medical sciences seems to be the next logical step. Such a transition will allow to connect different domains such as systems biology, systems medicine and medical informatics and might serve as a way to bridge the gaps between those scientific disciplines.

Contents

List of Figures

List of Tables

And what, Socrates, is the food of the soul? Surely, I said, knowledge is the food of the soul.

— *Plato*

1. A tale of knowledge lost

Motivation for investigating reusability and reproducibility of computational models - A tale of knowledge lost.

1.1. Euclides Danicus

Euclid of Alexandria, the "father of geometry", deduced the principles of what is today known as Euclidean Geometry based on a small set of axioms. In the first book of his series *Elements*, five axioms to carry out constructions in plane geometry are stated:

1. To draw a straight line from any point to any point.
2. To produce a finite straight line continuously in a straight line.
3. To describe a circle with any center and distance.
4. That all right angles are equal to one another.
5. That, if a straight line falling on two straight lines make the interior angles on the same side less than two right angles, the two straight lines, if produced indefinitely, meet on that side on which are the angles less than the two right angles.

Euclidean geometry is of a constructive nature, providing methods for construction using only a compass and an unmarked straightedge. Although Euclidean geometry has been used and worked on for centuries past, it remained an open question if a construction can be accomplished by either a compass or straightedge alone.

A brief history of Euclidean geometry. In his article "About the cover: Two theorems on geometric constructions", Alexanderson [2014] details on the history of solving the straightedge or compass question. The first widely available answer

to this question was given by Lorenzo Mascheroni in 1797, who proved that every basic construction of compass and straightedge can be carried out by compass alone [Mascheroni, 1797]. In 1833 Jakob Steiner proved a theorem, postulated earlier by Jean-Victor Poncelet, how to carry out every basic construction by straightedge only [Steiner, 1833]. This understanding of history stood until 1928, when a student in a bookshop in Copenhagen stumbled upon the book "Euclides Danicus" written by Mohr [1672] in a stack of used books.

Georg Mohr was a danish mathematician who studied and spend much of his lifetime in the Netherlands. In 1672 he published "Euclides Danicus" in Copenhagen and Amsterdam simultaneously. Apart from the Latin title, the language was Danish and Dutch. The book contained a proof of what was known since 1797 as Mascheroni theorem – in 1928 it was renamed Mohr–Mascheroni theorem.

Although the "Euclides Danicus" was published by Mohr in two languages, it's knowledge remained lost until Mascheroni independently discovered what Mohr had already proved a century ago. However, only due to fortunate circumstances, today the theorem is also rewarded to Georg Mohr.

Learning from history. Lost knowledge is common in human history. "Euclides Danicus" is just one example in line with the building of Egyptian Pyramids, Roman cement, Inca's talking knots, and an unknown number of ancient languages. In case of "Euclides Danicus", Alexanderson [2014] states:

> "There is a lesson here for all of us. In the 17th century the scientific *lingua franca* was clearly Latin and anyone interested in Mohr's result would have been able to read the text in Latin and recognize what he had done. But in Danish or Dutch? It didn't happen."

Georg Mohr's work was not findable – although he published his research, it was only accessible in Danish and Dutch. Consequently, the language barrier, nowadays one may call it a lack of standard, prevented "Euclides Danicus" from being interoperable and reuseable for the scientific community back then.

1.2. Systems Biology

Systems biology is an interdisciplinary field of studies at the intersection of biology, computer science, engineering, bioinformatics, physics, mathematics, medicine and others. Hence, it is hardly possible to provide a short and precise definition that everybody agrees on.

> "Systems biology is the science that studies how biological function emerges from the interactions between the components of living systems and how these emergent properties enable and constrain the behavior of those components." [Wolkenhauer, 2014]

Wolkenhauer states that systems biology is an holistic approach to understand complex biological systems and the idea that networks forming the whole of living organisms are more than the sum of their parts. The intention behind this postulate is to predict how biological systems change over time and under vacillating conditions. The innovation cycle and the interdisciplinarity of the field are illustraded in Figure

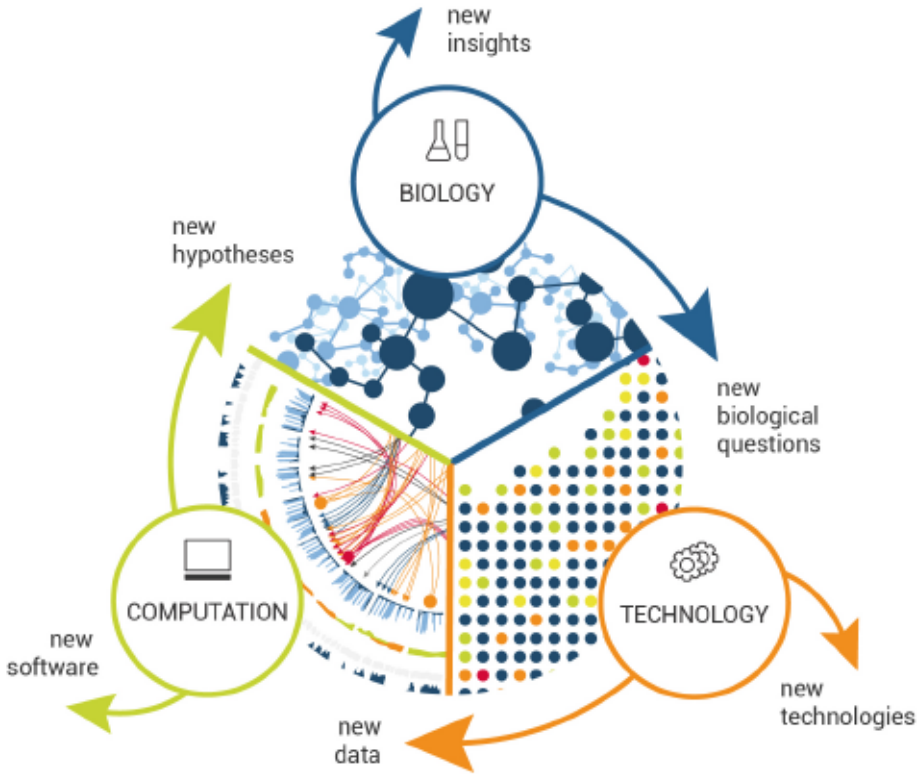


Figure 1.1.: *Cycle of innovation:* Systems biology is still a relatively new field of research. As such, solving new and challenging problems often requires the development of new methods and technologies and software to interpret and analyze and gain new data and insights. A key part of systems biology is dry-lab work, computationally solving biological problems using modelling.*

* This figure is taken from: https://omicscouts.com/media/files/blockcontent/2016-09/SystemsBiology.jpg

1.1. The computational part of this figure also contains an innovation cycle, regarding the scientific research process using modelling and simulation techniques. This process is depicted in Figure 1.2. As the example of "Euclides Danicus" shows, knowledge can be lost if certain criteria (i.e., findable, accessible, interoperable and reusable) are not met. Consequently, the research process described in Figure 1.2 must be

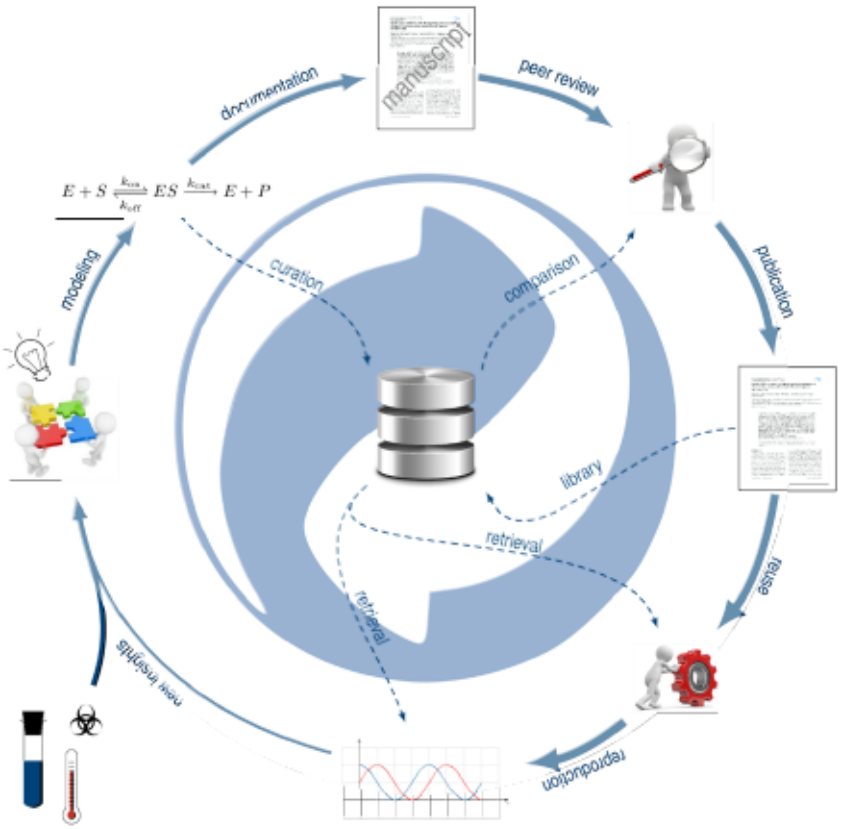


Figure 1.2.: *Model reuse and reproducibility:** With new insights, technology and data at hand, challenging hypotheses in systems biology can be solved by modelling and simulation. As this is inherently a research process, the resulting models and simulations are accompanied by a descriptive manuscript. Each step of the research process creates data that has to be managed in order to make models reusable and results reproducible.
* This figure is not published and only used in presentations. It was developed by Ron Henkel and refined by Martin Scharm.

backed by model and data management. The task of model management calls for an analysis of the challenges, and for the development of new ideas, methods and techniques to render models reusable and simulation results reproducible.

1.3. Aim of the Thesis

This thesis' aim is to describe and develop a model management in systems biology, applications to systems medicine are related but not in focus. More specifically, this thesis elucidates how models should be managed in terms of model storage and retrieval with respect to model provenance, reproducibility and reuse. In order to narrow down this general research objective to more specific research questions (simplified Research Questions are listed in Table 1.1) in Chapter 3.1, it is important

to introduce theoretical background (Chapter 2) on model storage, established standards and model reproduction and reuse in general. In almost the same manner, it is important to explain the research process (Section 1.3.1) behind this thesis. Both aspects, theoretical background and research process, ease the understanding of the thesis' context.

RQ#	Research Question
RQ1	How to establish a baseline for model management?
RQ2	How can heterogeneous, semi-structured model data be retrieved and ranked?
RQ3	How to trace a model over its lifetime?
RQ4	How to store a model and its accompanying meta-data?
RQ5	What model parts are important to elaborate on model similarity?
RQ6	What becomes possible once RQ1-5 are answered?

Table 1.1.: Simplified research questions (RQ). RQs will be deducted in more detail in Chapter 3.1

1.3.1. A brief History of the Thesis' Research Process

Research is not a linear process. Every research leads to new insights. New insights lead to new research or call for the refinement of previous research. The research process of this thesis is presented in Figure 1.3. Crucial parts of this research process are the identification of challenges and suggested solutions for model retrieval and ranking, model version control, model storage and the definition of model aspects. The challenges of *feature extraction* and *frequent structures in models* are not at the core of model management but became solvable because a model management was in place.

Challenges. In 2009 the community for systems biology models was relatively small, as was the number of models available. Popular repositories for models encoded in SBML [Hucka et al., 2003, 2010] and CellML [Cuellar et al., 2003, Lloyd et al., 2004] models were the BioModels Database [Li et al., 2009, 2010] and JWS [Van Gend et al., 2007] for SBML models and the Physiome Model Repository [Lloyd et al., 2008] for CellML models respectively [Köhn et al., 2009]. BioModels database contained 211 curated and annotated models, meaning the models are checked manually to reproduce the results described in a paper and enriched with annotations providing

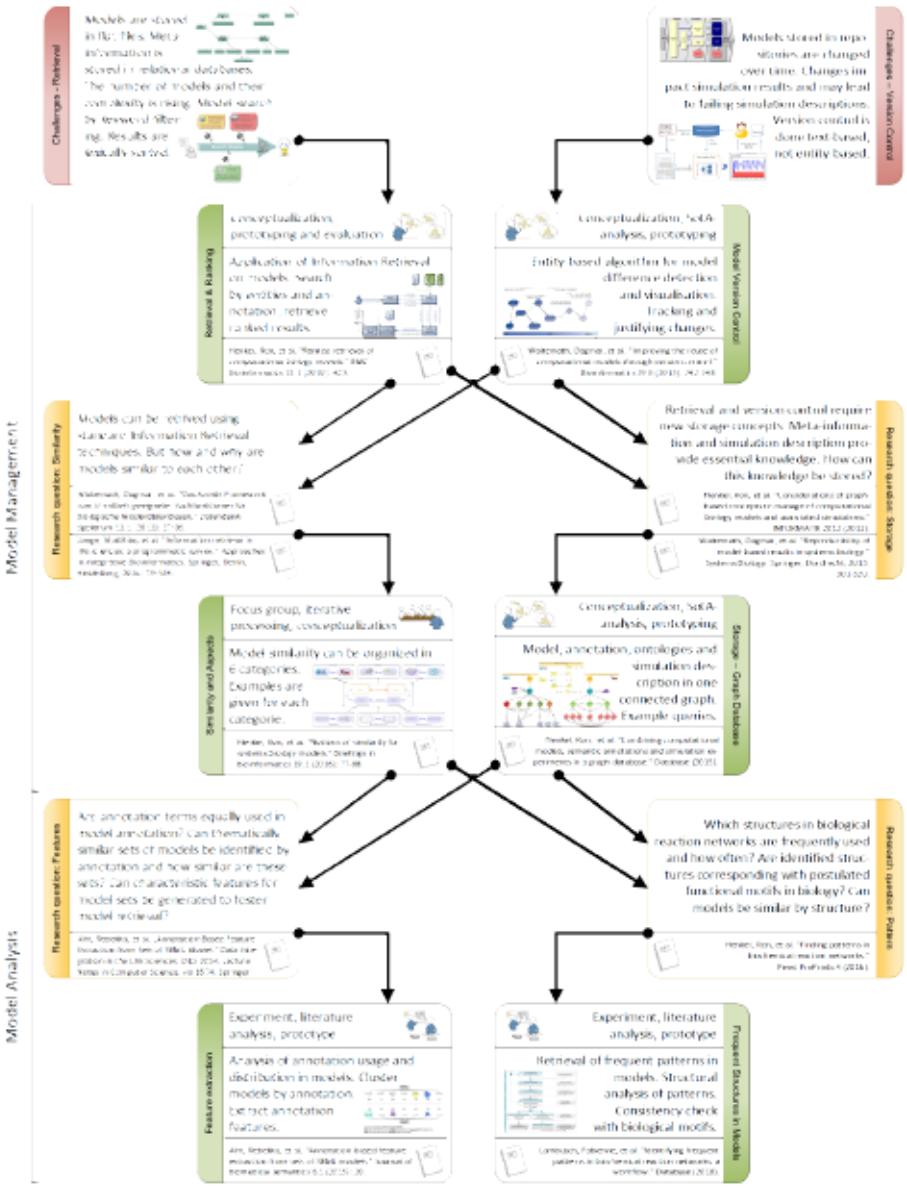


Figure 1.3.: *Research and knowledge development process:* Starting with the initial challenges (red boxes), this figure outlines the research activities and the development process leading to this thesis. Green boxes represent peer reviewed journal articles: the used research methods at the top, the main findings in the middle and the publication reference at the bottom. Yellow boxes represent upcoming research questions originating from the connected publication. In addition, at the bottom of each yellow box, publication references for preliminary considerations, surveys or statement papers are provided. All referenced publications are own contributions. (Figures in: "Challenges - Retrieval" box from Li et al. [2010], figures in: "Challenges - Version Control" box from Li et al. [2010] [top] and Waltemath et al. [2011b] [bottom])

specification for encoded entities. The Physiome Model Repository contained 386 models.

Models were mostly stored as files, with excerpts of meta-information in SQL Databases, or in version control systems like subversion. Retrieval was based on manually or semi-automatically provided keywords, created when adding a model to a repository. Typical categories for model retrieval, more specifically model filtering, were author name, publication title and journal, organism or model type. Results were ordered by a model ID, representing the chronological order of adding models to a repository. Lexical ordering by category was also an option.

Back then, BioModels Database had the most advanced retrieval techniques for biological models in comparison. It offered search for both, SBML structure elements and annotations (only by using an ontology ID). Given a keyword or an ID, the system returned not only models that contain the keyword in the XML code but also models annotated with the provided ontology ID, respectively. However, obvious drawbacks were that BioModels Database search (1) is restricted to SBML models, (2) does not rank search results and (3) does not support entity based version control of models [Köhn et al., 2009].

It became apparent that a sophisticated search engine and similarity measures for systems biology models would greatly improve model retrieval and help researchers to find, reproduce and reuse models [Henkel et al., 2009]. In addition, a system to track model changes on entity level instead of line-based model code tracking would foster model provenance and further increases reproducibility and reusability.

Retrieval & Ranking. To tackle the shortcommings in model retrieval, a search approach for systems biology models was developed, including a prototype implementation added in BioModels Database [Henkel et al., 2010]. The approach is based on existing retrieval and ranking methods from Information Retrieval (e.g. Boolean and vector space model). It incorporates the capability to not only search the model but also the textual descriptions behind annotations used in a model.

The introduced approach and implementation were, at this time, the first application of Information Retrieval techniques on model search in computational systems biology. Having an implementation in BioModels Database at hand, it shows that the approach is beneficial for searching for relevant models and for retrieving ranked results. This was a first step towards better search capabilities in model databases and expected to have a positive effect on the reuse of existing models.

Model Version Control. After discussing the model ranking and retrieval approach with experts in the fields and extending the approach to include simulation studies,

new insights on version control shortcomings became available. The Simulation Experiment Description Markup Language (SED-ML, [Waltemath et al., 2011b,c], a specification to describe how a model should be simulated, started to play an increasing role. Simulation studies, incorporated in the aforementioned model search, allowed a user to retrieve models by keyword. If available, models are delivered enriched with simulation studies. SED-ML addresses specific entities within a simulated model by an XPath expression. The model to be simulated is provided by direct linking to a model in a repository, e.g. BioModels Database. Interestingly, many simulation runs failed as the repository silently and without explanation changed the models (e.g. to correct a spelling error of an entity) but published the model using the same link. It became apparent that models should be put under a sophisticated version control system tailored towards the specification a model is encoded in and providing justification for the changes made. At the time model version control achieved using line-based algorithms such as longest common subsequence. As a model inherently represents a graph of interacting entities, a line based algorithm is hardly feasible to keep a models provenance and development [Henkel et al., 2011]. This insight lead to a proposal on how to improve model reuse through version control [Waltemath et al., 2013a]. The basic idea is to do the version control entity-based and allow to justify each change.

Storage - Graph Database. With the lessons learned from connecting simulation studies and models, specifying a version control system, a rising number of semi-automatically created models (each bearing countless annotations), the idea of representing and storing a model and all accompanying information as an interconnected graph became apparent [Henkel et al., 2012b]. To underpin the usefulness and uniqueness of the graph approach, insights about the reproducibility of model based results [Waltemath et al., 2013b] were published. The proposed approach is based on a graph database and represents models (SBML and CellML), simulation studies (SED-ML) and annotations as an interconnected graph of nodes and relationships. In combination with an adapted information retrieval system it was, for the first time, possible to query entities and structure at the same time [Henkel et al., 2015]. The system is capable of answering complex queries, using information from and connections between models, annotations and simulation descriptions. Such a query is, for example: "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*". Since 2013 the

Physiome Model Repository uses this system as a retrieval back-end for the advanced search of CellML models.

Subsequently, the idea of representing and storing biological information in a graph database was adapted to accommodate pathway information [Touré et al., 2016] encoded in the Systems Biology Graphical Notation (SBGN, [Le Novère et al., 2009]).

Similarity and Aspects. The understanding what makes models similar to each other depends on the point of view and research field. For example, a modeler may focus on the entities and the structure they are encoded in, while a biologist may be more interested in the information provided by biological and chemical annotations and a computational biologist may value the description of dynamics produced by simulating the model. To accommodate those needs, a framework for model similarities was proposed [Waltemath et al., 2011d]. With broader view, provided by survey of information retrieval in life sciences [Lange et al., 2014], the question how and why models are similar in general became apparent. This lead to a publication defining model similarity on a broader spectrum [Henkel et al., 2016a]. Six aspects, potentially relevant for model similarity were defined: underlying encoding, references to biological entities, quantitative behaviour, qualitative behaviour, mathematical equations, and parameters and network structure. For each aspect, existing similarity measures, tools and areas of application are provided and illustrated by example.

1.3.2. A research environment to thrive in

In order to foster model reuse and reproducibility, to construct a model repository, to develop a model version control, to design a model retrieval or define model aspects it is necessary to analyse **what information** is available, **how it is encoded**, **how can it be accessed** and **how it can be processed**.

A model's habitat. A model in order to be reproducible, as Figure 1.2 shows, can not stand on its own. It is accompanied by and linked to a variety of additional information and data. This additional information and data, not to mention the model itself, is encoded by different standards, stored in repositories, accessed by application interfaces, simulated by different solvers, and linked using semantic technologies. Altogether, a habitat where a model exists in is formed. The following Chapter 2 explains this habitat and thereby introduces relevant standards, repositories, software and data-sources. The definition of a models habitat is relevant to define criteria for model storage, version control and retrieval with respect to information and data

accompanying a model. For this Chapter are of relevance:

Waltemath et al. [2013b] Waltemath, D., Henkel, R., Winter, F., & Wolkenhauer, O. (2013). *Reproducibility of model-based results in systems biology.* In Systems Biology (pp. 301-320). Springer Netherlands.

Lange et al. [2014] Lange, M., Henkel, R., Müller, W., Waltemath, D., & Weise, S. (2014). *Information retrieval in life sciences: a programmatic survey.* In Approaches in Integrative Bioinformatics (pp. 73-109). Springer Berlin Heidelberg.

Henkel [2020] Henkel, R. *Biomedical repositories for simulation studies.* In Systems Medicine: Integrative, Qualitative and Computational Approaches. Elsevier Netherlands.

The greatest responsibility of the planner and architect, I believe, is the protection and development of our habitat.

— *Walter Gropius*

2. A models habitat

Next to existing standard formats for model and simulation description encoding, this chapter describes the theoretical background for model repositories, databases and tools necessary for model storage, reproduction and reuse. In addition, it is exemplified how meta-information is made available and linked to a model or simulation description as defined by so called minimum information guidelines.

Every organism, even the tiniest ones, need an environment to live and thrive in. All together, exchange formats, simulation descriptions, ontologies, meta-information guidelines and repositories set boundaries for modelling, simulation and model reuse - or in other words, form a habitat a model can reside in.

This chapter contains parts of publications from Henkel [2009, 2020], Lange et al. [2014], Touré et al. [2016] & Waltemath et al. [2013b]. It explains the systems biology and computer science background relevant for this thesis.

2.1. Reproduction keeps a habitat alive

A computational model of a biological system is an abstract representation of the living system, simplified by a number of assumptions, and instantiated with a certain set of parameter values. Computational models of biological systems can be diverse in scale and complexity, ranging from 'omics'-scale (i.e., modeling whole genomes and proteomes) to modeling small sub-circuits of a network [Ferrell, 2009].

During development, these models are often written in software tools such as Matlab, or are directly programmed in languages such as R, Python, Java or C. One way of studying the models and thereby obtaining results is through simulation. A simulation mimics the behavior of the system, for example by calculating the changes in concentration of a particular entity over time. It is important to understand

that simulations are not part of a model, but can be applied on a model to study its behavior. To reproduce a model-based result, one does therefore not only need the model itself, but one must also have available information about the simulation that was run on the model to show a desired effect, for example oscillating behavior. When models and associated simulations need to be exchanged (e.g., together with a publication, or in large-scale collaborative projects) model databases help distributing code. Here, specifically designed, computer-readable standard representation formats allow for model exchange across different software tools and projects. Exporting model code in a standard format is a common practice in many modeling communities, for example when developing biochemical models, or in physiology.

Computational encodings of models are already available from open databases and repositories such as BioModels [Chelliah et al., 2013, 2015, Glont et al., 2017, Le Novère et al., 2006, Li et al., 2010] and Physiome Model Repository [Lloyd et al., 2008, Sarwar et al., 2019, Yu et al., 2011, 2015]. The encoding of a model (e. g. in SBML [Hucka et al., 2003, 2010] or CellML [Cuellar et al., 2003, Lloyd et al., 2004, 2008] format) contains the model structure and initial parameterization. To understand the intention behind a model one may read the reference publication, study related models, explore the information available from the repository or database, or investigate simulation experiments performed on the model. In addition, if the model is sufficiently annotated with information from biology ontologies one can infer semi-automatically the model's scope and level of detail together with the implied assumptions. Annotations link to entries in openly available biology ontologies such as the Gene Ontology [Ashburner et al., 2000, Bernasconi and Masseroli, 2019] or ChEBI [Degtyarenko et al., 2008, Hastings et al., 2015]. These links are stored using semantic technologies such as the Resource Description Framework (RDF) [Lassila et al., 1998b]. For example, a modeler can semantically enrich a model with entries from the Gene Ontology, ChEBI or other bio-ontologies. Thereby, supporting the interpretation of the model's biological scope. A model with a detailed explanation of all involved species can be faster understood; a model with associated simulation experiments and simulation results can be better interpreted, and modifications on the simulation settings can be easier reconstructed. The provision of annotated models in standard format is particularly important as the modeling process more often than not demands to incorporate previously obtained findings from collaborators and literature, as has been exemplified with the yeast metabolic network model [Herrgård et al., 2008]. Standards for model code ease the exchange of models across work groups, as they make model code usable in different software environments.

The exchange of computational models enables the reproduction of other researchers' works, the study of results that have already been obtained for a given biological question, and even the adjustment or extension of models to test own hypotheses. In summary, to reproduce model-based results in computational biology,

1. Models should be encoded in standard formats;
2. Meta-information should be provided to help understanding the models' intention; and
3. Associated simulation experiments should be encoded in standard formats.

All information must be made available through open repositories to foster reuse and consequently allow for result reproducibility.

2.2. Standards

Extensive standardisation efforts in computer-aided exploration of biomedical systems led to the foundation of COMBINE [Hucka et al., 2015, Myers et al., 2017], a global network that coordinates the development of standards and common terminologies for modeling and simulation in computational biology. An active community with a strong wish to improve reproducibility of model-based research output drives the further development of these standards and advertises the COMBINE-compliant publication of scientific results.

2.2.1. Exchange Formats

SBML. The Systems Biology Markup Language (SBML) is a community effort encompassing researchers and software developers from different institutes. It is a standard representation format for the description of biochemical reaction networks, including cell signaling pathways, but also metabolic pathways, gene regulation networks etc. To date SBML has been adopted by more than 296 software systems[1] from simulators to modeling tools and databases. As such it can be considered the most successful standard in the field so far [Li et al., 2010]. SBML's major revisions are called levels. A level represents substantial changes to the composition and structure of the language. The current language specification is SBML Level 3 Version 2 Core (Release 2)[2]. With Level 3, SBML has switched to a modular, plug-in

[1] For a list of compatible systems, please refer to: `http://sbml.org/SBML_Software_Guide`
[2] Please refer to: `http://sbml.org/Documents/Specifications#SBML_Level_3_Packages`

type of language specification: In addition to the core language, so-called packages are developed for specific modeling needs, e. g., a spatial package or annotation package [Waltemath et al., 2011e, Zhang et al., 2020]. A core SBML model incorporates the following main parts [Hucka et al., 2010, 2018]:

Function definition: A named mathematical function available for use in the model,

Unit definition: A named new unit of measurement,

Compartment: A container for species location, either representing physical structures or not. It is assumed to be a well-stirred one,

Species: A pool of any kind of entities of the same kind,

Parameter: A quantity with a symbolic name (constants or variable, global or local),

Initial assignment: A mathematical expression defining the initial condition of the model,

Rule: A mathematical expression defining how to calculate the value or the rate of change of a variable,

Constraint: A mathematical expression computing a true/false value from model variables, parameters and constants,

Reaction: A statement with an associated rate expression and describing a process that might change the amount of one or more species,

Event: A statement describing instantaneous, discontinuous change in one or more variables when a condition is triggered.

Species define pools of biological entities within an SBML model that may reside in an enclosed environment, a compartment. Each species has an id, a location within a compartment, and an initial amount or concentration. The optional name helps when displaying the model code to the users. A reaction defines species as reactants (input), modifiers and product (output), thereby forming a reaction network. SBML reuses the MathML standard to encode the mathematical expressions contained in the model, including the kinetic rates, equations, or function definitions. The SBML species is additionally annotated with meta-information in RDF format, following the MIRIAM guidelines [Le Novère et al., 2005].

The MIRIAM guidelines (see 2.2.2) require each model to present a minimum set of annotation to provide detailed information about a model and a model's constituents. This annotation refers to an external resource and is required to unambiguously relate a piece of knowledge to a model constituent. In case of SBML, a triplet of

{data type, identifier, qualifier} describes the referenced information, while the data type has to be encoded as Unique Resource Identifier (URI). It is possible, but not mandatory, to use qualifiers refining the connection between the model constituent and the piece of knowledge. Table 2.1[3] shows some important qualifiers.

Qualifier	Description
is	The biological entity represented by the model component is the subject of the referenced resource
isPartOf	The biological entity represented by the model component is a physical or logical part of the subject of the referenced resource
hasVersion	The biological entity represented by the model component is a version or an instance of the subject of the referenced resource
isHomologTo	The biological entity represented by the model component is homologous, to the subject of the referenced resource

Table 2.1.: Biology qualifiers from BioModels.net used by MIRIAM

CellML. CellML is a model description language for specifying and exchanging biological processes. The current version is CellML 1.1[4]. It has a modular structure and focuses on the mathematical description of biological processes. The explicit representation of modularity and the extensibility of the language both permit the description of a range of cellular and sub-cellular systems, including biochemistry, electrophysiology, system physiology and the mechanics of the intra-cellular environment [Lloyd et al., 2004]. CellML, in contrast to SBML, provides generic components which may be abstract groupings, or be representing biological entities and physical compartments. A CellML model is built as a network of connections between self-contained elements [Cuellar et al., 2003]:

Model: The CellML root element, contains all the following elements,

Component: Smallest functional unit of a model, contains the variables and mathematics to describe the behavior of the subsystem,

Connection: Connects components to each other, and maps variables in one component to variables in another,

Import: Allows for import of further valid CellML models,

[3]Qualifiers taken from BioModels.net, `https://co.mbine.org/author/biomodels.net-qualifiers/`

[4]CellML version 2.0 is currently in draft status: `https://www.cellml.org/specifications`

Unit: Allows for the definition of units, apart from standard units already provided; every variable and number has to have a unit assigned,

Group: Allows for the definition of logical (encapsulation) and physical (containment) relationships between components to form hierarchical structures (i. e., a tree of components linked by parent-child relationships of the same type).

Components are additionally annotated with meta-information in RDF format, but the annotation does not necessarily follow the MIRIAM guideline [Wimalaratne et al., 2009]. Similarly to SBML, an initial value is assigned to a component. A components interactions are specified using variable declarations and MathML, thereby forming a reaction network.

SED-ML. To enable result reproducibility in the life sciences, model authors should ideally provide Simulation Experiment Description Markup Language (SED-ML) [Waltemath et al., 2011b,c] files together with their model code, describing how to reproduce the presented simulation results. End-users can thereupon run the simulation experiments in simulation software of their choice. They might furthermore share their own simulation experiment descriptions by exporting SED-ML descriptions from their simulation tool. SED-ML is agnostic about the underlying model representation formats and the software tool that ran the experiment. The model variables that a SED-ML model needs to be aware of are addressed directly by XPath; the only limitation is for the model code to be XML. The current version of SED-ML is Level 1 Version 3 [Bergmann et al., 2018].

The SED-ML format is built of five main elements: (1) the models used in the experiment; (2) the simulation algorithms and their configurations; (3) the combination of algorithm and model into a numerical experiment; (4) post-processing of results; (5) and output of results. Each SED-ML file holds a list of references containing all models used in the simulation experiment. Ideally, the reference is an unambiguous link to a model in an open model repository, making code retrieval easier. If adjustments of model parameters are necessary before simulation (pre-processing), they can be encoded in SED-ML as well. The format furthermore allows storing changes in the model's XML structure. All changes are defined by linking to a particular model entity with XPath, which is the standard technology to address nodes within an XML file. A SED-ML file furthermore contains all information on the type of simulation (e. g., time course), the solver that has been used (e. g., CVODE) and the additional settings of that solver. The Kinetic Simulation Algorithm Ontology (KiSAO) [Courtot et al., 2011] has been specifically designed to structure

the knowledge about simulation algorithms used in computational biology, and about their characteristics. Post-processing steps necessary in between simulation and output of the experimental results are encoded in a specific XML structure within the SED-ML file. Finally, the type of output can be stored, including information about the assignment of plotted entities to the axes of the plots. To date, several communities establish SED-ML as a standard to exchange simulation setups; the main supporters so far are the SBML and CellML communities. Software support for SED-ML has been implemented in simulation tools, libraries, validation tools, and a SED-ML visual editor [Adams, 2012].

NeuroML. The NeuroML [Cannon et al., 2014] standard is the equivalent in computational neuroscience of what SBML is for computational biology. The NeuroML community aims at making models in NeuroML format available from ModelDB [McDougal et al., 2017]. NeuroML Version 2.0 contains specifications about cell, cell morphology, networks, synapses, ion channels and inputs. In addition, NeuroML can hold RDF annotations and simulations can be defined using SED-ML.

SBGN. The Systems Biology Graphical Notation (SBGN) [Le Novère et al., 2009] is a visual representation of biological networks. It is common in systems biology and fills the previous gap of standardized visual representations for biological networks.

SBGN is composed of a set of three complementary languages: Process Description (PD), Activity Flow (AF) and Entity Relationship (ER). PD shows "all the molecular processes and interactions taking place between biochemical entities, and their results" [Le Novère et al., 2009]. It creates detailed SBGN maps by representing hierarchical structures and biological complexes. AF "shows only influences such as 'stimulation' and 'inhibition' between the activities displayed by the molecular entities" [Le Novère et al., 2009]. It is the most elementary of the SBGN languages. Finally, ER shows all "influences of entities upon the behaviour of others", ignoring any temporal aspect [Sorokin et al., 2015]. SBGN diagrams are drawn using three sets of standardized glyphs for the three SBGN languages. SBGN maps are represented in an XML-based format: the SBGN-ML [Van Iersel et al., 2012], which is both human-readable and machine-readable. They can be exported from software tools such as SBGN-ED [Czauderna et al., 2010] (a VANTED add-on [Rohn et al., 2012] with SBGN support) or CellDesigner [Funahashi et al., 2008].

Additional exchange Formats. SBOL is an open standard for the representation of *in silico* biological designs, developed by the SBOL development group. It provides a data format composed of genetic vocabulary terms and schematic glyphs to graphically depict genetic designs. The current specification of SBOL is Version 2.3 [Madsen et al., 2018].

BioPAX [Bader and Cary, 2005] is an exchange format to enable integration, exchange and analysis of biological pathway data. It uses OWL as underlying specification. The current version is BioPAX Level 3 [Demir et al., 2010].

2.2.2. Recommendations and guidelines

MIBBI - the Minimum Information for Biological and Biomedical Investigations Taylor et al. [2008], is a community driven approach to develop and make available minimum information checklists in the area of life sciences. The primary purpose of minimum information checklists is to guide researchers in reporting their experiments, ensuring the most relevant information to understand, reproduce and reuse scientific findings are provided along with their experiments, studies or models. While MIBBI has a broad scope in life sciences and biomedical sciences, two guidelines, MIRIAM and MIASE, are of particular interest for this thesis.

MIRIAM guidelines. The Minimum Information Requested in the Annotation of Models (MIRIAM) [Le Novère et al., 2005] describes the minimum of information necessary to be provided together with a computational model in order for it to be useful to others. Both the SBML and CellML format follow these guidelines.[5] Annotations provide links to information that is not covered in the XML representation, but must be made available. These links point to entries in external biology ontologies, databases, controlled vocabularies or classifications. A better human-understandable interpretation of SBML or CellML encoded models can be generated by resolving the model annotation. For example, given a species or component named *cdc2k* and annotated with a MIRIAM identifier[6]. The annotation can be resolved using the MIRIAM query services. The annotation points to the entry with ID P04551 in the UniProt ontology, representing *Cyclin-dependent kinase*, a detailed description of the biological entity is provided.

As a scope Le Novère et al. [2005] defines the terms quantitative and encoded models:

[5]CellML models hosted in the Physiome Model Repository are, however, not necessarily curated.
[6]Cyclin-dependent kinase at Uniprot: `http://identifiers.org/uniprot/P04551`

Definition 1 (Quantitative Model) *A quantitative model is a formal model of a biological system, based on the mathematical description of its molecular and cellular components, and the interactions between those components.*

Definition 2 (Encoded Model) *An encoded model is a mathematical model written in a formal machine readable language, such that it can be systematically parsed and employed by simulation and analysis software without further human translation.*

In addition, MIRIAM defines the following set of rules that should apply to an encoded model.

1. A public, machine-readable format is mandatory for encoding models. For example SBML meets this requirement.
2. The encoded model must pass a validation regarding to the language used for encoding.
3. A single reference description must be related with the model. The description should provide the expected results produced by the model.
4. Relevant simulation results given in the description must be reproduceable
5. The biological process specified in the description has to be the related to the model structure. The constituent of the description and the model do not need to be related one to one, but they should have an analogous structure.
6. All quantitative attributes of the model have to be defined. Values for all initial conditions have to be defined or referenced.

Furthermore, MIRIAM requires that a minimum set of annotations have to be included with a quantitative model.

1. The name of the model.
2. A citation of the reference description with which the model is associated.
3. Name and contact information of the creators.
4. The date and time of creation as well as last modification.
5. A statement about the terms of distribution applying to this model.

MIASE guidelines. MIASE (Minimum Information About a Simulation Experiment) [Waltemath et al., 2011a] defines, in a textual description, the information that is to be provided together with a model in order to ensure the reproducibility of the experiments run on that model. SED-ML is the computational format to store and exchange that information.

Waltemath et al. [2011a] defines the set of rules to be applied to simulation descriptions as follows:

1. All models used in the experiment must be identified, accessible, and fully described.
    a) The description of the simulation experiment must be provided together with the models necessary for the experiment, or with a precise and unambiguous way of accessing those models.
    b) The models required for the simulations must be provided with all governing equations, parameter values, and necessary conditions (initial state and/or boundary conditions).
    c) If a model is not encoded in a standard format, then the model code must be made available to the user. If a model is not encoded in an open format or code, its full description must be provided, sufficient to re-implement it.
    d) Any modification of a model (pre-processing) required before the execution of a step of the simulation experiment must be described.
2. A precise description of the simulation steps and other procedures used by the experiment must be provided.
    a) All simulation steps must be clearly described, including the simulation algorithms to be used, the models on which to apply each simulation, the order of the simulation steps, and the data processing to be done between the simulation steps.
    b) All information needed for the correct implementation of the necessary simulation steps must be included through precise descriptions or references to unambiguous information sources.
    c) If a simulation step is performed using a computer program for which source code is not available, all information needed to reproduce the simulation, and not just repeat it, must be provided, including the algorithms

used by the original software and any information necessary to implement them, such as the discretization and integration methods.

d) If it is known that a simulation step will produce different results when performed in a different simulation environment or on a different computational platform, an explanation must be given of how the model has to be run with the specified environment/platform in order to achieve the purpose of the experiment.

3. All information necessary to obtain the desired numerical results must be provided.

   a) All post-processing steps applied on the raw numerical results of simulation steps in order to generate the final results have to be described in detail. That includes the identification of data to process, the order in which changes were applied, and also the nature of changes.

   b) If the expected insights depend on the relation between different results, such as a plot of one against another, the results to be compared have to be specified.

It is an immanent requirement to be able to evaluate the quality of scientific activity. MIRIAM and MIASE both focus on life science applications. If both guidelines are applied to models and simulations, respectively, it allows to reproduce simulations and thus understand and verify a models findings. It also improves sharing of models and simulation procedures and thereby promoting collaborative development and reuse of models.

2.3. Repositories, Databases and Tools

2.3.1. Repositories and Databases

Several journals ask for submission of the model code in an open model repository when submitting a manuscript: While the model needs not be publicly available at the time of manuscript submission, it will be necessary to provide access to the model via a global identifier. That way, the reviewers can download the model and check the model outcomes described in the manuscript.

Open model repositories collect, curate and then publish models and associated simulation studies, experimental data, and other relevant information to reuse the model. Published models get a permanent identifier (PURL) for reference. Some

repositories offer starring-systems to document the level of curation or the level of completeness of models. At the minimum, open model repositories offer access to the curated, standards-compliant model code. Some repositories offer a richer set of associated data, including the simulation instructions for recreation of the plots displayed in a paper, or access to the experimental data that led to the formulation of the hypotheses. A third type of relevant meta-data are semantic annotations – they give meaning to the pure syntactical description of a model.

2.3.2. Resources for curated models

Open model repositories offer an infrastructure for scientists to publish their model code. Models are curated (checked for validity and reproducibility) and get a persistent identifier to guarantee long-term access and citeability. Furthermore, publication in a model repository means online promotion, higher access rates (and thus visibility) and automated linkage with other relevant resources, including bio-ontologies, related models and simulation studies, or experimental data.

The available repositories were originally associated with a model representation format, e. g., BioModels (see 2.3.3 was mainly focussed on SBML, while the Physiome Model Repository (see 2.3.3 was considering CellML models. This distinction is still visible, but not a limitation. For example, the FAIRDOMHub (see 2.3.5) model repository, which is based on SEEK (see 2.3.5) has models in many different formats; BioModels accepts not only SBML models, but a variety of different open standards (and even non-open formats such as MATLAB may be submitted).

2.3.3. Repositories for biochemical kinetic models

BioModels

BioModels is a repository of freely accessible model code [Chelliah et al., 2013, 2015, Glont et al., 2017, Le Novère et al., 2006, Li et al., 2010]. It is an open-source project for commercial and academic use. BioModels[7] accepts models submitted by modelers (e. g., for reference in a publication), but it also imports models from collaborative model repositories such as the CellML Model Repository (see 2.3.3). While the main focus is on SBML-encoded models, the submission of other formats is also supported; the files will then be transformed using specifically developed converters, e. g., [Gómez et al., 2016, Rodriguez et al., 2016].

[7]Biomodels: `https://www.ebi.ac.uk/biomodels`

BioModels was launched at the EMBL-EBI in 2005, and since then its content grew steadily. Today, BioModels offers 831 manually curated models and 1129 non-curated models. In addition 143,070 models automatically generated from pathway resources are available in a separated branch [Büchel et al., 2013]. However, BioModels does not only grow in terms of the amount of models, but also with regard to the variety of modeled entities and the size of encoded reaction networks. Curated models have been checked by professionals with regard to their syntactical validity (compliance with one of the SBML levels), semantically enriched (addition of links to bio-ontologies), and tested for reproducibility of at least one result from the reference publication. One also refers to these enhancements as "aiming to make a model MIRIAM-compliant", meaning that the models meet the minimum requirements for the annotation (documentation) of model code in computational biology [Le Novère et al., 2005].

BioModels provides several services, including model curation and annotation [Juty et al., 2015]. Models can easily be accessed through the web interface or a REST-API, and even be simulated using one of the embedded simulation tools. The SBML file history is tracked using a version control system. Meta-data that is available from the model file includes information on the submission and modification dates of a model file, authors' information, references and annotations encoded in a MIRIAM-compliant manner. As Glont et al. [2017] states, an effective search is crucial for any repository to encourage model reuse and foster reproducibility. BioModels search system is based on the EBI search engine [Park et al., 2017], which allows to not only search for model metadata but also takes into account the model file itself and uses semantic enrichment to further improve search results. Search results can be filtered on various model characteristics, i.e. model format, organism, ontology entries or modelling approach.

The content of BioModels is also available as a linked data set using Semantic Web technologies [Wimalaratne et al., 2014]. Here, SBML encoded models are accessible as RDF-resources via a SPARQL-Endpoint[8]. The SPARQL-Endpoint exposes the models as a dataset interoperable with other semantic web resources, e.g. ontology entries directly referenced in models.

[8]SPARQL-Endpoint: `https://www.ebi.ac.uk/rdf/services/sparql`

Physiome Model Repository

The Physiome Model Repository [Lloyd et al., 2008, Sarwar et al., 2019, Yu et al., 2011, 2015] is the standard repository for CellML models at different stages of curation. The repository is designed to provide a powerful platform, helping scientists to reuse and reproduce models, while fostering collaboration and the exchange of simulation experiments. The Physiome Model Repository[9] implements a model management system based on the content management system Plone and the version control system Git. It can be used either as an online application or as a standalone application. Available models cover different biological processes, including signal transduction pathways, metabolic pathways, electrophysiology, immunology, cell cycle, muscle contraction and mechanical models [Lloyd et al., 2004]. These biological processes are described by mathematical equations along with all variables, and description.

The Physiome Model Repository contains about 619 model exposures encoded in the CellML format (as of December 2019). A CellML exposure consists of a model and its associated documentation and meta-information. An exposure is a particularly interesting and also well-documented version of a workspace that has been approved for publication by a curator. A workspace (a private place within a users Git repository) can be shared with other users allowing for easier collaboration during model development. In addition workspaces can be build in a hierarchical manner, reflecting the idea of modularity in CellML models. Models in the Physiome Model Repository are browsable by different (physiological) categories, including cell cycle, signal transduction, or metabolism. A system-wide full-text search is offered that permits free-text search. In addition, model browsing based on Ricordo [de Bono et al., 2011] is available.

The Physiome Model Repository also provides extensive web services[10]. Open Authorization (OAuth) standard is used to establish the authentication and to manage access privileges. Model resources are provided, for example, as JSON-Objects or as RDF, accessible via a SPARQL endpoint [Munarko et al., 2019, Sarwar et al., 2019].

JWS online

JWS online [Olivier and Snoep, 2004, Van Gend et al., 2007] is a freely availabe web service for creating, simulating and storing models. It aims to provide the systems

[9] Physiome Model Repository: `https://models.physiomeproject.org/cellml`

[10] Web Services Guideline: `https://aucklandphysiomerepository.readthedocs.io/en/latest/webservice.html`

biology community with access to a web based database of curated models that can be parameterised and simulated directly in a web browser. In addition to simulation, JWS online is capable of directly linking simulation data and experimental data files stored in the FAIRDOMHub [Peters et al., 2017, Wolstencroft et al., 2016]. Providing model, simulation and experimental data in such a manner improves the reuse and reproducibility of scientific work and improves the publication process.

The JWS Online Model Database[11] is part of the JWS Online Simulator, a web-based simulator for biochemical kinetic models. The simulator supports time evolutions, steady-state simulations, structural analysis, metabolic-control analysis, parameter scans, and reaction plots.

The model repository serves as the maintainer for kinetic models that can be interactively run online. It supports filtering for SBML models by a limited number of characteristics, including the author, publication title and journal, organism or model type. A web-based tool offers a searchable categorization of models in the repository, distinguishing, for example, between cell cycle models and metabolism. In terms of searching, JWS online uses filtering by Name, Process, Organism and Type. JWS Online is used as a model repository within collaborative European projects. For example, it is integrated with the SEEK platform [Wolstencroft et al., 2015, 2016] which is used by FAIRDOM and de.NBI consortia to improve and support model management.

JWS online provides a powerful REST-API[12] capable of searching, down- and uploading, analysing and exporting and simulating models and simulation experiments, respectively.

Since 2017 JWS online offers a simulation database[13] [Peters et al., 2017], containing curated simulation experiments of stored models. Next to listing and linking all available simulations per model, the simulation database allows for a "one-click figure reproduction" by directly simulating the model and reproducing the referenced manuscript figure.

BiGG

BiGG Models[14] is a platform for whole genome-scale metabolic network reconstructions [King et al., 2015]. It contains more than 75 manually curated and published genome-scale metabolic networks. A unique standardised identifier is provided for

[11] JWS Model Database: `https://jjj.bio.vu.nl/models/`
[12] JWS Online documentation: `https://jws-docs.readthedocs.io/`
[13] JWS Simulation Database: `https://jjj.bio.vu.nl/models/experiments/`
[14] BiGG Models: `http://bigg.ucsd.edu/`

every network and model, respectively. BiGG Models includes a REST-API and provides programmatic access to JSON encoded data.

2.3.4. Repositories for neuroscience models

ModelDB

ModelDB, initially started in 1996 [Peterson et al., 1996], has been growing into a valuable resource of open model code in computational neuroscience [McDougal et al., 2017], where authors can share their models without restriction on, for example, simulator choice or modeling topic. ModelDB currently contains 1490 models that can be searched, viewed, download, simulated and shared by scientists.

Models stored in ModelDB[15] can be searched and discoverd based on the meta-information which had been provided at model upload and curation process. Here, a tag based categorization (e. g., model type, brain region, cell type, genes or model concepts) is used to qualify a model accordingly. ModelDB, in contrast to the databases mentioned above, does not put restrictions on the format of submitted model code, and it does not require standard formats. Consequently, ModelDB can handle different types of model code very flexibly. When downloading a model from ModelDB, the user must comply with the data and tools necessary to run the model. For example, ModelDB offers a large number of models dealing with synaptic plasticity. Some of them require Matlab to execute m-files, others demand XPP or C++. Although many models have an auto-run feature linking to the NEURON simulator [Hines, 1993], a program for simulation of nerve equations, users may have to set up a whole computational environment in order to reuse the model code. To ease and track model reuse, ModelDB automatically indicates when a file is reused in a different model. This indication works regardless of model context and allows comparisons across models. Furthermore, ModelDB explicitly fosters this way of reusing model code. Reuse is able at different levels, e.g. code snippets, model components such as ion channels or and whole models. ModelDB also fosters reproducibilty by the curation process [McDougal et al., 2016], that ensures that models are fit to run on as many platforms as possible and on different simulator versions.

[15] ModelDB: `https://senselab.med.yale.edu/modeldb/`

Open Source Brain

Open Source Brain[16] is a platform for sharing, viewing, analyzing, and simulating standardised models from different brain regions and species [Gleeson et al., 2019]. Bowser-based simulation and visualization of model structure and parameters are a key-feature [Quintana et al., 2015]. Infrastructure and tools for collaborative interaction are provided by the underlying Git repository. Open Source Brain is suited for high level models of cognitive processes and consciousness to low level models of signal transduction at synapses preferably encoded in simulator independent formats such as NeuroML and PyNN.

2.3.5. General repositories for scientific data

SEEK

The SEEK[17] platform [Wolstencroft et al., 2015], as a web-based resource, is a suite of tools that allows to share scientific research data. It is not only designed to host models and simulations, but also to manage and organise different research data from the domain of systems biology. Using the ISA (Investigation, Study, Assay) infrastructure, it preserves associations between heterogeneous data, along with provenance information. Within SEEK, ISA has been extended and is configurable to allow the structure to be used outside of Biology. An investigation is a high level description of a research area, e.g. a research project, that can contain several studies. A study is a precise hypothesis that is to be tested and validated by wet-lab experiments or modelling and simulation studies. Each study can contain different assays. An assay an individual experiment or modelling analysis that contains and produces specific data.

SEEK incorporates semantic technology allowing sophisticated queries over the data, yet without getting in the way of the users. Its key features are a data catalogue, version control, model and simulation annotation, extensive and well-defined access control, publication integration, and a person index.

SEEK does not necessarily store all data, e.g. proteomics files, itself but operates with semantic links to incorporate large datasets. An advantage of SEEK is, that all information necessary to understand and recreate an experiment is available at a single place. The JWS Online model simulator (see 2.3.3) is directly accessible from within SEEK and used for simulation studies linked to a model within an assay.

[16] Open Source Brain web site: `http://www.opensourcebrain.org/`
[17] Seek4Science: `https://seek4science.org/`

Seek also uses RightField templates [Wolstencroft et al., 2011]. RightField is an open-source tool for adding ontology term selection to Excel spreadsheets. By using such templates, scientists are encouraged to provide meta-data (e.g., for models and simulations studies) according to the JERM [Wolstencroft et al., 2013] compliance and standardisation effort.

SEEK provides a RestFul API. In addition, the underlying semantic web resources extract and serve SEEK data items as RDF and XML. SEEK RDF enables rich semantic queries and integration with other RDF resources (e.g. EBI linked data set [Wimalaratne et al., 2014]). The SEEK Platform has been adopted by large consortia such as SysMO (Systems biology for Micro-organisms)or VLN (virtual Liver Network).

FAIRDOMHub

The FAIRDOMHub[18] [Wolstencroft et al., 2016] provides an open SEEK instance. As of today it offers 505 models and 2512 data files organised in 281 investigations. FAIRDOMHub combines and offers tools and services to manage, store, and publish data, models, and operating procedures. This includes the SEEK platform and openBIS. FAIRDOMHub is based on the FAIRDOM is designed as a joint action of ERA-Net (ERASysAPP) and European Research Infrastructure (ISBE). It aims to improve and establish data and model management for systems biology and follow the idea of FAIR (Findable, Accessible, Interoperable and Reusable) data management.

2.3.6. How to find a model of interest?

If a scientist is looking for a model of particular interest, the first step is to use the retrieval, filter end discovery systems provided by the repositories described in this Chapter. Mostly, the author's name, a PubMedId, the organism or the biological process of interest is enough information to query the repositories and retrieve the results, which is, preferably, a model.

For example, if a scientist is interested in a model by Tyson from 1991 simulating the cell cycle as described in the publication Tyson [1991], a variety of retrieval options are available on FAIRDOMHub. The scientist may use the FAIRDOMHub search for the keyword "cell cycle" and filter by the category model. A search for the keyword "Tyson" also retrieves the desired model. If the scientist's intention is to find a model for a specific publication, a search for the PubmedID (in this chase

[18]FAIRDOMHub: `https://fairdomhub.org/`

"1831270") retrieves the publication. When viewing the retrieved publication item, a graph shows all investigations, studies, assays, model and data files related to this publication.

Once the scientist finds the desired model, in this case the Tyson1991 - Cell Cycle 6 var model[19] that was uploaded from BioModels[20] to FAIRDOMHub, the retrieved model and accompanying data can be studied. With a click on the "Simulate Model on JWS" button, the model is transferred to the JWS Online simulator (see 2.3.3). Figure 3a of the publication [Tyson, 1991] can be reproduced by adjusting the initial concentrations for $M = 0.75$ and $YP = 0.25$, set the end time for the time evolution to 100 and observe the species M and the assignment rule YT. As the retrieved model is shipped with a SEDML file, it is also possible to upload the SEDML file to the JWS Online Simulation Database or a compatible simulation tool[21] of the scientist's choice to simulate the model directly.

2.3.7. Tools

A model could not be constructed, simulated, annotated, visualized, analyzed, shared or be published without the respective tools. The following paragraphs describe the most common editors and simulators as well as software necessary to annotate and shre simulation studies. The specific tools and software mentioned in the following are only a

Construction and Simulation. For construction and simulation of systems biology models a variety of tools and libraries are available. Copasi [Bergmann et al., 2017a, Hoops et al., 2006] is a widely used software for constructing, simulating and alaysing biochemical networks. It supports SBML and SED-ML import and export. SED-ML Webtools [Bergmann et al., 2017b] is an online application to edit, simulate and validate SED-ML. It handles SBML as well as CellML models. OpenCor [Garny and Hunter, 2015] is a widely used editor and simulation tool for CellML encoded models. It also provides validation and visualization capabilities and SED-ML support. In addition to model construction and simulation, most tools are capable of a multi-format export.

[19] Link to model file on FAIRDOMHub: `https://fairdomhub.org/models/196`
[20] Link to model file on BioModels: `https://www.ebi.ac.uk/biomodels/BIOMD0000000005`
[21] SED-ML Web Tools: `http://sysbioapps.dyndns.org/SED-ML_Web_Tools`

Reproducible Simulation Studies. The COMBINE Archive [Scharm et al., 2014] is a zip-like, meta-data enriched format to bundle all files relevant for a simulation study in computational biology. It was developed to foster the exchange of reproducible simulation studies. A COMBINE Archive may include, among other resources, models and their graphical representations, data sets such as simulation results or wet lab data, simulation experiment descriptions, standard operating procedures and publications. Every file in the COMBINE Archive can be annotated with supplementary meta-data. To keep the provenance of created archives, meta-data such as creation and modification date, as well as the creator of a file is mandatory. A COMBINE Archive can be created from a SEEK investigation or uploaded to JWS, guaranteeing the contained simulation can be reproduced. Next to SEEK and JWS, COMBINE Archives can be read by many other software tools.

Research Objects (ROs, [Bechhofer et al., 2013]) are designed to pack, publish, preserve and share scientific data among researchers. Similarly to COMBINEArchive, ROs are compressed data-structures containing all relevant files to reproduce a scientific study. A manifest within the RO contains the relevant metadata and describes the content of the package.

BioSimulators[22] [Shaikh et al., 2022] is a central registry for simulation tools. BioSimulators is build on the usage of standards such as CellML, SBML, SED-ML and the COMBINE archive format and in addition incorporates validation tools for simulation studies to ensure standard and simulation consistency.

Resources to provide persistent identification. The Minimum Information Required in the Annotation of Models Registry [Juty et al., 2012, Wimalaratne et al., 2015] provides identifiers for data used in the biomedical domain as a location-independent service. The registry is based on a collection of data catalogues, linking each catalogue to an individual namespace. This allows to generate Uniform Resource Identifiers (URIs) to precisely identify any record within a collection. The Identifiers.org service[23] is the entry point for manually or automatically accessing the registry.

[22]BioSimulators: `https://biosimulators.org`
[23]Identifiers.org service: `http://identifiers.org`

2.4. Controlled vocabularies and Ontologies

One way of providing the information required by MIRIAM is to refer to knowledge encoded in biological ontologies. For example, some of the Unique Resource Identifiers used to refine a constituent point to ontologies like "Systems Biology Ontology" (SBO, Bernasconi and Masseroli [2019], Le Novère et al. [2007]). The advantage of pointing to an ontology rather than pointing to a plain definition is the gain in information. Knowing about the position of the refined constituent within an abstract concept can be valuable.

In the area of computer science an ontology is a formal representation of a set of concepts within a domain plus the relationships between those concepts. In the area of biology ontologies are used to provide controlled vocabularies and their relations. A general definition is given by Gruber et al. [1993], stating that an ontology is an explicit specification of a conceptualization.

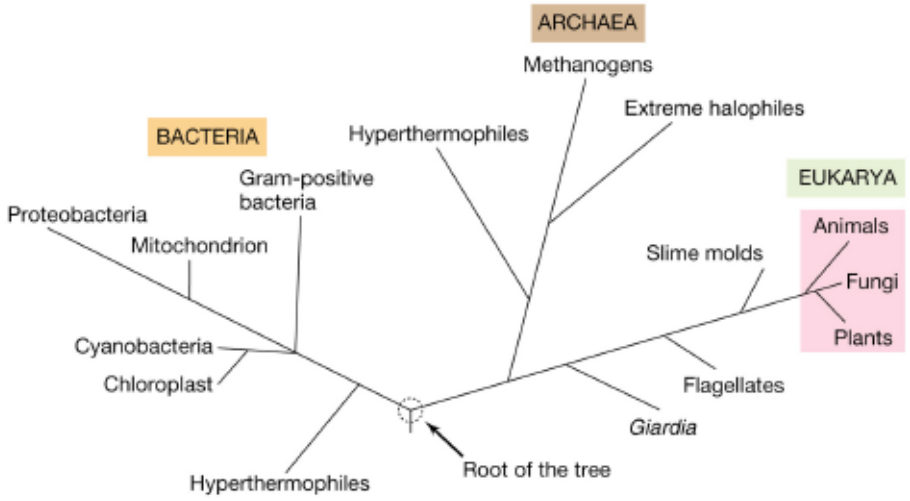


Figure 2.1.: "Phylogeny of Life" from [Brock et al., 1974]

The most obvious usage is to represent the taxonomy of species as shown in figure 2.1 (a taxonomy is a special ontology type). Other ontologies provide controlled vocabularies for molecular functions or entities, biological processes or cellular components. A variety of ontologies with biological or chemical vocabulary can be used within an annotation as an external resource. The following examples of ontologies are possible sources of annotations for model constituents:

Chemical Entities of Biological Interest (ChEBI [Degtyarenko et al., 2008, Hastings et al., 2015]) is a freely available dictionary of molecular entities focused

on chemical compounds. The molecular entities might be products of nature or synthetic products interacting in or with living organisms.

Gene Ontology (GO [Ashburner et al., 2000, Consortium, 2018]) provides controlled vocabularies in different areas of biology. The ontology for Molecular Function describes activities at the molecular level. Biological Process (BP) describes higher level processes where more then one molecular function is participating. Cellular Component (CC) describes macromolecular complexes and sub cellular locations. Sequence Ontology (SO) describes sequence features.

Universal Protein Resource (UniProt [Apweiler et al., 2004, Consortium et al., 2008, 2018]) is a resource for protein sequence and functional annotation. It comprises four components: UniProt Knowledgebase (UniProtKB) for protein information, UniProt Archive (UniParc) for protein sequences, UniProt Reference Clusters (UniRef) to cluster related sequences, and UniProt Metagenomic and Environmental Sequences (UniMES) for metagenomic and environmental data.

Kyoto Encyclopedia of Genes and Genomes (KEGG [Kanehisa et al., 2011, 2016, Ogata et al., 1999]) consists of 15 main databases that can be categorized into systems information (PATHWAY, BRITE, MODULE, DISEASE, DRUG and ENVIRON), genomic information (ORTHOLOGY, GENOME and GENES) and chemical information (COMPOUND, GLYCAN, REACTION, RPAIR, RCLASS and ENZYME). KEGG provides and integrates resources including genomic, chemical and systemic functional information. In addition, KEGG links gene catalogs from sequenced genomes to cell, organism and ecosystem functions.

Systems Biology Ontology (SBO [Bernasconi and Masseroli, 2019, Le Novère et al., 2007]) describes entities used in computational modeling. SBO defines a set of complemental concepts, used to specify the type or role of a component in a model. It contains seven branches: systems description parameter, participant role, modeling framework, mathematical expression, occurring entity representation, physical entity representation, and metadata representation. SBO is an open, community developed ontology.

Kinetic Simulation Algorithm Ontology (KiSAO [Courtot et al., 2011]) As MIASE states, it is necessary for a simulation experiment to unambiguously define the proper simulation algorithm. KiSAO is tailored towards describing and structuring simulation algorithms. It structures simulation algorithms in a

hierarchical manner and, in addition, links algorithms to their characteristics (e. g. if a solution is approximate or exact).

TErminology for the Descriptions of DYnamics (TEDDY [Courtot et al., 2011]) describes the observed behavior of a simulated model in a machine-readable manner (e. g. oscillation or stable limit cycle). It comprises four branches: behavior characteristic, temporal behavior, functional motif and behavior diversification.

Just Enough Results Model (JERM [Wolstencroft et al., 2013]) is a minimal descriptor of necessary information to ensure reproducibility of biology experiments.

Figure 2.2 shows controlled vocabularies and ontologies used in SBML encoded models, it also provides an idea on the variety of model annotation using external resources.

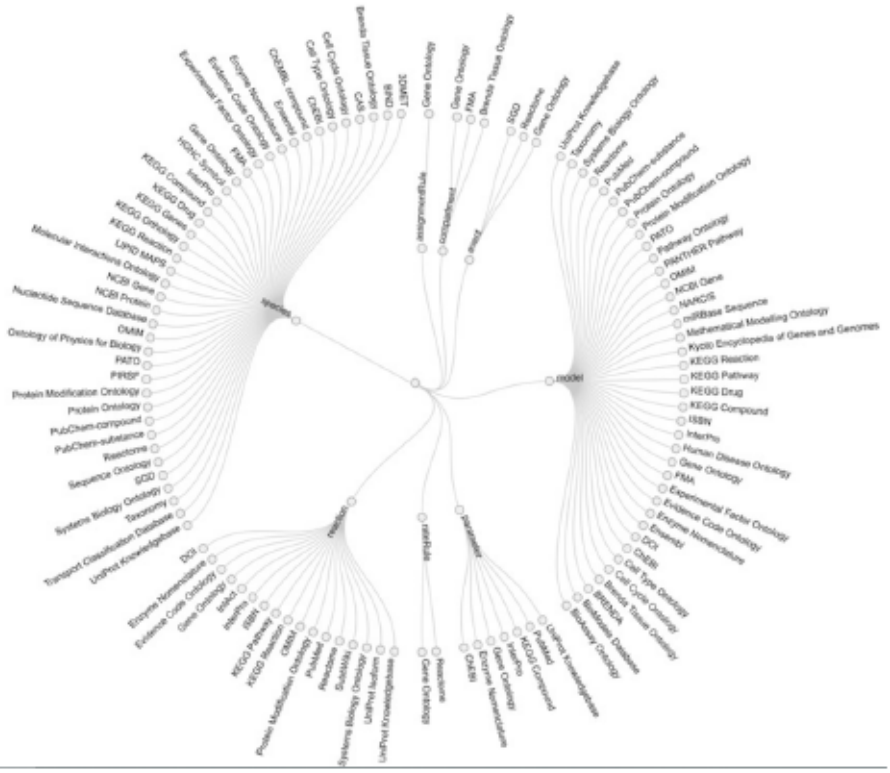


Figure 2.2.: *Annotations in BioModels:* This Figure from Stanford et al. [2019] shows external resources (outer circle) used to annotate models from BioModels during the curation process. The inner circle state the elements of an SBML model as explained in Section 2.2.1.

2.5. Reusability & Reproducibility

Stating that "Only reproducible results are of significance to science.", Waltemath and Wolkenhauer [2016] elucidate how modeling standards, software and initiatives are essential to reproduce results in the systems biology and systems medicine domain. Using the example of cancer research, they show that there is a huge lack if it comes to the reproducibility of cancer studies. *Bayer* published an analysis, stating that 24 of 67 findings in cancer studies could not be reproduced as stated by Prinz et al. [2011]. Similarly, an analysis by *Amgen* points out the irreproducibility of 47 out of 53 oncological studies (see Begley and Ellis [2012]). And recently, Tiwari et al. [2021] showed that 49% of the published models in BioModels were not directly reproducible. Subsequently, Waltemath and Wolkenhauer [2016] identify four route causes for irreproducibility:
(1) A lack of standards in data generation, leading to integration and comparability issues.
(2) A lack of quality and quantity of data, reducing the finding's significance.
(3) A lack of openness and long-term access, limiting the availability of models and data.
And *(4) a lack of transparency*, caused by unstated assumptions regarding the data and the model. The lack of transparency also includes missing tools, workflows or parameter values necessary to understand the scientific findings.

To overcome the reproducibility hurdles a variety of efforts is necessary. Here, Waltemath and Wolkenhauer [2016] distinguishes between infrastructure and model management efforts.

Infrastructure. Infrastructure subsumes software as well as initiatives and community networks aiming to ease reproducibility in systems biology and systems medicine. The *Reproducibility initiative*[24] is a collaboration of PLOS ONE (open-access scientific publisher), Science Exchange[25] (marketplace for commercial contract research organization) and Figshare[26] (digital repository for researchers). This initiative offers researchers means to an independent validation of their scientific findings.

Community networks are vital for every scientist. The "COmputational Modeling in BIology NEtwork"[27] (COMBINE) is an initiative and collaboration network hosting

[24]Reproducibility initiative: `http://validation.scienceexchange.com/#/reproducibility-initiative`
[25]Science Exchange: `https://www.scienceexchange.com/`
[26]Figshare: `https://figshare.com/`
[27]COMBINE: `http://co.mbine.org/`

open standards for modeling in systems and computational biology. FAIRDOM is a European project for data and model management services. Its core principle is to make models and data *f*indable, *a*ccessible, **i**nteroperable and *r*eusable. The "German Network for Bioinformatics Infrastructure"[28], de.NBI, is an infrastructure project providing a plethora of bioinformatics services, for example consulting and teaching, data storage, software tools and workflows, or data analysis. de.NBI itself is part of ELIXIR,[29] a European initiative and distributed infrastructure for life sciences.

Regarding software to enhance reproducibility, Waltemath and Wolkenhauer [2016] list a variety of open-source, open-access tools and workflows, allowing researchers to create, curate, simulate, integrate and store their data and models.

Model Management. Models are only useful if they can be accessed and if it is clearly documented how the results can be reproduced [Le Novère, 2006]. While initiatives and community networks provide standards and tools to foster model reproducibility, equally important, model management calls for accessibility, documentation and curation. Waltemath and Wolkenhauer [2016] defines four parts of model management:

(1) Repositories and management systems. This provides access to curated models. This goes along with the necessity to search for, and retrieve models.

(2) Curation. A thorough and independently curation ensures the validity of the model in terms of syntax, semantic, pragmatic and scientific content. Curation includes annotation and thus allows to unambiguously identify variables and parameters in terms of their chemical or biological meaning.

(3) Documentation and reporting, is the documentation written according to community agreed standards and does the model reflect the results described by the related paper. This also calls for Minimum Information guidelines [Taylor et al., 2008] to be followed.

(4) Sharing reproducible simulation studies. Although raw simulation data might be shipped along with the model, the model results are not necessarily reproducible. With just raw simulation data at hand, one still needs to verify under what parameterization the model produces this specific data. A simulation study, along with the model, is used to describe in detail the conditions (e. g. parametrization, the variables to be observed, simulation algorithm, plot specification). Chapter 2.2 and

[28] de.NBI: `https://www.denbi.de/`
[29] ELIXIR: `https://www.elixir-europe.org/`

2.3 explain in more detail repositories, database, guidelines, and exchange formats relevant for this thesis.

Responsibilities. Model reusability and reproducibility, although narrowed down to the context of systems biology, is still a broad field. Enhancing model reproducibility and reusability can be done on different levels. Politically, by directing funds to research calls. Publisher-wise, by demanding certain standards for models alongside publications. Community-wise, by agreeing on a shared set of guidelines and standards. And personally, by enforcing and obeying the aforementioned.

I have no illusions concerning the precarious status of my tales and do not expect to become a serious competitor of my favorite weird authors.

— H.P. Lovecraft

3. Contributions to reuse and reproducibility of computational biology models

Although a lot has been achieved, with respect to model reproducibility and reusability, since the emergence of the field of systems biology – many problems are still ahead, even some we are still unaware of. This thesis, however, can only analyze and describe a selected part of an often much bigger problem. Context of this thesis is model management in systems biology, applications to systems medicine are related but not in focus. More specifically, this thesis elucidates how models should be managed in terms of model storage and retrieval with respect to model provenance, reproducibility and reuse. As already mentioned by Waltemath and Wolkenhauer [2016], a model can only be reproduced and reused if it is curated, documented and its simulation studies are available. Naturally, this thesis takes these surrounding information about a model into consideration as well.

Furthermore, this thesis uses the term "model" in a specific context. Hearing the term model, a biologist might think about a model organism (e. g. *Drosophila melanogaster* or *Escherichia coli*), a chemist visualizes a molecular model and a physicist imagines the standard model of particle physics. In this thesis, we focus on systems biology models. Systems biology models can be based on a number of mathematical formalisms [Le Novère, 2015]. Here, the choice of a particular approach largely depends on the type of question asked and on the data available [Wolkenhauer, 2014]. Further on, the term "model" refers to systems biology models, encoded in a machine-readable standard exchange format, preferably annotated with ontological

concepts and shipped along with a reproducibility study.

3.1. Research questions and Thesis Structure

Waltemath and Wolkenhauer [2016] stated that model management is the key point to foster model reuse through model repositories, standards, minimum information guidelines and simulation studies as driving factors. In addition, Wilkinson et al. [2016] define the FAIR principles findable, accessable interoperable and reusable as guidelines for scientific data management and stewardship. This thesis focus is to **establish a baseline for a FAIR model management (RQ1[1])** specifically in the fields of model ranking & retrieval Henkel et al. [2010], model version control Waltemath et al. [2013a] and model storage Henkel et al. [2015].

An overview on how the topics model ranking & retrieval, model storage and model version control can be mapped to model management and the FAIR principles is shown in Figure 3.1.

The term *model*, even if narrowed down to the interpretation as stated above, is still very broad. The view and interpretation as of what a model is and what the important model parts are, still varies with the viewer being a mathematician, chemist, biologist, physician, physicist or computer scientist. Therefore, Henkel et al. [2016a] defines different model aspects. Similarity techniques and measures are assigned to those aspects if possible and research gaps, e.g. similarities for mathematical expressions, are explained.

3.1.1. A model's habitat

The Chapter 1.3.2 already introduced the idea of a model's habitat - an environment for model to live, thrive and "reproduce" in. Chapter 2 described the necessary background: standards, exchange formats, repositories, ontologies and minimum guidelines. This knowledge is used in the following chapters to explain model search (or how to find inhabitants), model version control (in other words, an inhabitants life cycle), model storage (how to do the accommodation for inhabitants) and model aspects (also called a models nature).

[1] RQ* denotes a research question

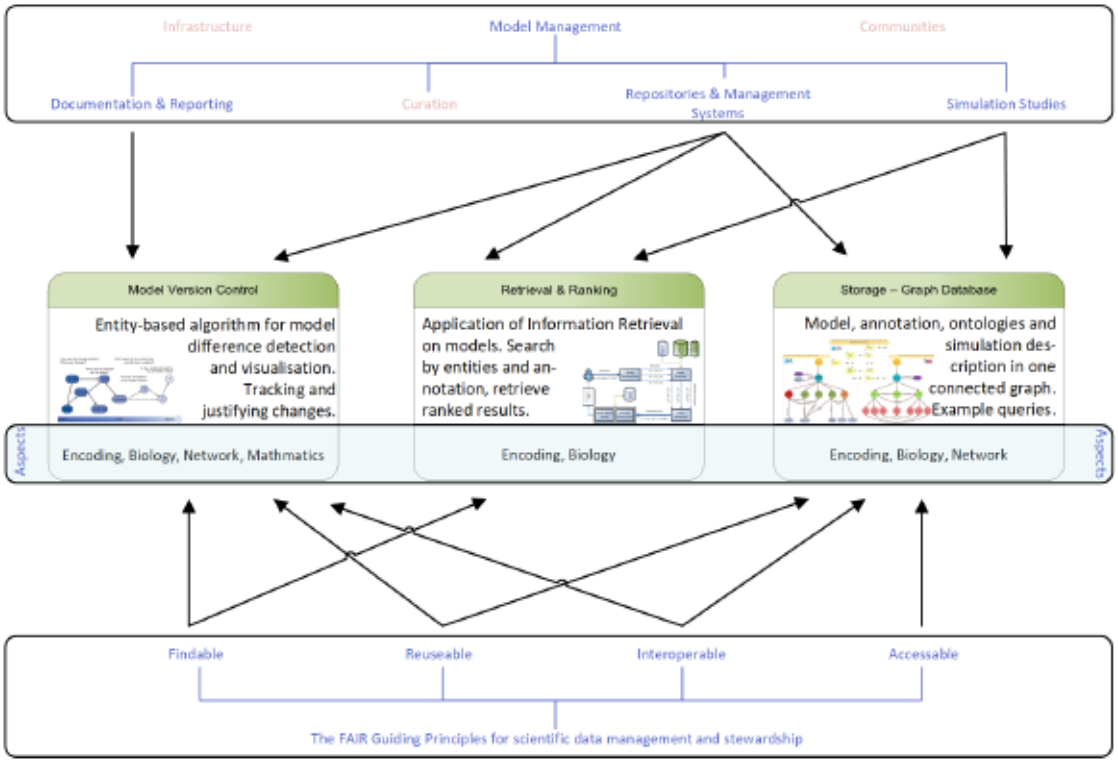


Figure 3.1.: *Contributions to Model Management and the FAIR principles:* This figure illustrates the contributions of thesis to model management as defined by Waltemath and Wolkenhauer [2016] and the Fair principles Wilkinson et al. [2016]. In addition, the relevant model aspects as defined in Henkel et al. [2016a] are highlighted. As an example, model management requires repositories and management systems. A contribution towards this requirement is the graph-based storage concept for models, simulation studies and accompanying data (MaSyMoS, Henkel et al. [2015]). Relevant model aspects affected are model encoding, biology and network. As for the FAIR principles, this concept provides accessibility for models and simulation studies.

3.1.2. Finding the inhabitants.

The idea to create a model retrieval engine was the initial driver for this thesis. When the first version of model retrieval was designed, the only way to search for models was by using SQL or rely on the build in search engine of content management systems - surely not the optimal solution given the knowledge about a models habitat, accompanying information and vast annotation possibilities. The research question here is, **how heterogeneous, semi-structured information** enriched by meta-data such as annotation can **be retrieved and ranked (RQ2)**. Chapter 4 explains the idea of using a model index and a semantic index combined to enable model ranking and retrieval by keywords only present in the textual representation of a models annotation.

Henkel et al. [2010] Henkel, R., Endler, L., Peters, A., Le Novère, N., & Waltemath, D.

(2010). *Ranked retrieval of computational biology models.* BMC bioinformatics, 11(1), 423.

3.1.3. An inhabitants life cycle.

The ability to retrieve a model from a repository is the first step towards model reuse. However, to be able to understand and reuse models as well as to reproduce their results, documentation and reporting is crucial [Waltemath and Wolkenhauer, 2016]. An important part of a model's documentation is provenance. Oxford dictionary defines provenance as "A record of ownership of a work of art or an antique, used as a guide to authenticity or quality"[2].

During its life cycle, a model is subject to a characteristic set of changes, reflecting upon the model's updates from its creation to curation, publication and possible later reuse. A prerequisite for the study of a model's history is the availability and understanding of all significant model versions. In this context, provenance means that every significant **model version should be accessible (RQ3)**, along with the information **what was changed, by whom and why**. The Chapter 5 explains how version control can improve model reuse, it is based on the following publication:

Waltemath et al. [2013a] Waltemath, D., Henkel, R., Hälke, R., Scharm, M., & Wolkenhauer, O. (2013). *Improving the reuse of computational models through version control.* Bioinformatics, 29(6), 742-748.

3.1.4. Accommodation for inhabitants

As aforementioned, different model repositories and databases are already in place. However, mostly their storage concepts support only restricted types of queries and not all data inside the repositories can be retrieved. Chapter 6 elucidates a graph database storage concept that meets this challenge. The created database application, called MaSyMoS (Management System for Models and Simulations) **reflects a models' structure, incorporates semantic annotations and simulation descriptions (RQ4)** and ultimately connects different types of model-related data. The publication relevant for this Chapter is:

Henkel et al. [2015] Henkel, R., Wolkenhauer, O., & Waltemath, D. (2015). *Combining computational models, semantic annotations and simulation experiments in a graph database.* Database, 2015, bau130.

[2]Definition according to Oxford Dictionary: `https://en.oxforddictionaries.com/definition/provenance`

3.1.5. A model's nature.

After defining a models habitat, analyzing life cycles and building accommodation for the inhabitants, it is necessary to understand **what the important bits and pieces of a model are (RQ5)**. Chapter 7 explains different points of view, so called "aspects", of a model. Six aspects potentially relevant for model understanding and comparison are defined: underlying encoding, references to biological entities, quantitative behavior, qualitative behavior, mathematical equations and parameters, and network structure. To keep up with the analogy of the model's habitat, these aspects can be understood as a model's nature. This in depth understanding of a model's nature is essential to define how a model should be stored and what model aspects are necessary for a ranked model retrieval. The relevant publication with this author's contribution is:

Henkel et al. [2016a] Henkel, R., Hoehndorf, R., Kacprowski, T., Knüpfer, C., Liebermeister, W., & Waltemath, D. (2016). *Notions of similarity for systems biology models.* Briefings in Bioinformatics, bbw090.

Knowing now what to store in a model repository the next puzzle to be solved is **how to store it**. The discussion on how to store identified information is presented in Chapter 6.

3.1.6. Habitat status report

Having a semantically integrated model storage and retrieval at hand the question **what else can be achieved with the information (RQ6)** immanent. Chapter 8 contains contributions and further developments that are relevant for a model's nature or habitat, describe alternative approaches to model storage and retrieval, or originating from putting MaSyMoS into practice. As this Thesis' focus is on fostering FAIR model management by model retrieval, model version control and model storage, the following research works are not explained in depth.

Model retrieval and ranking with graphs

Changing the key technology concepts for storage and defining how a model is connected to the accompanying information inherently demands for an adaptation of the model retrieval and ranking. The idea of model retrieval and ranking as described in Chapter 4 is based on the idea of having a stand-alone index without a database back-end. With the developments described in Chapter 6, the premise changed fundamentally. A model retrieval can now be enhanced to also incorporate

model structure and accompanying data into a model search. Section 8.1.1 describes the adaptations applied to make a model retrieval based on models stored as graphs.

Proceedings in model version control

Chapter 5 outlines conceptual requirements for model version control, methods for identification and justification of changes, and a prototype implementation. Section 8.2 briefly describes the further algorithm development and refinement, the COMODI ontology to specify changes in a model, and how to keep track on the evolution of computational models in a database.

Model analysis with graphs

Most models are curated and annotated. Annotations are a reliable source of knowledge linking model entities to concepts in established ontologies. Hence, the question arises if this rich source of knowledge can be used to characterize a model or a set of models? Section 8.3.1 elucidates methods for annotation-based **feature extraction** from model sets and is based on:

Alm et al. [2015] Alm, R., Waltemath, D., Wolfien, M., Wolkenhauer, O., & Henkel, R. (2015). *Annotation-based feature extraction from sets of SBML models.* Journal of biomedical semantics, 6(1), 20.

Another way to characterize models is by looking at common and shared patterns of their reaction networks. Means to identify, find, and understand biologically relevant motifs helps to reuse model parts. Section 8.3.2 introduces a tool chain that identifies reoccurring **patterns in biochemical reaction networks**:

Henkel et al. [2016b] Henkel, R., Lambusch, F., Wolkenhauer, O., Sandkuhl, K., Rosenke, C., & Waltemath, D. (2016). *Finding patterns in biochemical reaction networks.* PeerJ PrePrints, 4, e1479v2.

Lambusch et al. [2018] Lambusch, F., Waltemath, D., Wolkenhauer, O., Sandkuhl, K., Rosenke, C., & Henkel, R. (2018). *Identifying frequent patterns in biochemical reaction networks: a workflow.* Database, bay051, 2018.

Biological pathways.

More focused on graphically displaying pathways, STON (SBGN TO Neo4j) is a tool to read, store, query and display pathways encoded in SBGN (Systems Biology Graphical Notation). It focuses on metabolic, signaling and gene regulatory pathways. Areas of application are to identify subnetworks, link pathways and networks over

different granularity levels, and identify common patterns between pathways. Section 8.4 describes STON, based on:

Touré et al. [2016] Touré, V., Mazein, A., Waltemath, D., Balaur, I., Saqi, M., Henkel, R., Pellet, J., & Auffray, C. (2016). *STON: exploring biological pathways using the SBGN standard and graph databases.* BMC Bioinformatics, 17(1), 494.

CovidGraph.

With the upcoming of the Covid-19 pandemic in 2020, a new demand for model reuse became apparent. Helping researchers to quickly and efficiently find their way through COVID-19 datasets is a valuable contribution to fighting the pandemic. This research describes CovidGraph, a Neo4j graph database users can use to explore papers, patents, treatments and medications covering the family of corona viruses. In addition to literature and data, biomedical models from the MaSyMoS database were incorporated [Henkel et al., 2015] and connected to their entities in biology - namely genes and proteins and their function to already existing data domains, thus spanning a network of unparalleled size and knowledge. This can be seen as a first approach to bridge the gap between systems biology and medical informatics in this particular field. Section 8.5 describes CovidGraph, is based on:

Gütebier et al. [2022] Gütebier, L., Bleimehl, T., Henkel, R., Munro, J., Müller, S., Morgner, A., Laenge, J., Pachauer, A., Erdl, A., Weimar, J., Walther-Langendorf, K., Vialard, V., Liebig, T., Preusse, M. Walthemath, D. & Jarasch, A. (2022). *CovidGraph: A Graph to fight COVID-19.* Bioinformatics, Bioinformatics, Volume 38, Issue 20, Pages 4843–4845.

3.1.7. Conclusion

Chapter 9 comes back to the story of Euclides Deanicus and finishes the "tale of knowledge lost". Section 9.1 reflects on the research questions as stated in this chapter and summarizes what has been achieved and how those questions can be answered. Section 9.2 elucidates limitations of the research presented and the answered research questions, while Section 9.3 provides a perspective on the way ahead and concludes this thesis.

But do you know that, although I have kept the diary for months past, it never once struck me how I was going to find any particular part of it in case I wanted to look it up?

— *Bram Stroker, Dracula*

4. Finding the inhabitants

Ranked retrieval of Computational Biology models [Henkel et al., 2010] - the ability to retrieve a model from a repository is the first step towards model reuse. Consequently, the idea to create a model retrieval engine was the initial driver for this thesis. The following chapter describes the first application of ranking and retrieval methods, common in document retrieval, to the field of computational systems biology models.[1]

Research question: The study of biological systems demands computational support. If targeting a biological problem, the reuse of existing computational models can save time and effort [Waltemath and Wolkenhauer, 2016]. Deciding for potentially suitable models, however, becomes more challenging with the increasing number of computational models available, and even more when considering the models' growing complexity. Firstly, among a set of potential model candidates it is difficult to decide for the model that best suits ones needs. Secondly, it is hard to grasp the nature of an unknown model listed in a search result set, and to judge how well it fits for the particular problem one has in mind. On a more abstract level, models consist of heterogeneous, semi-structured information enriched by meta-information such as annotation. The research question is how to retrieve and rank such kind of information.

Result: This Chapter describes an improved search approach for computational models of biological processes. It is based on existing retrieval and ranking methods from Information Retrieval [Baeza-Yates and Ribeiro-Neto, 1999]. The described

[1] see Appendix A.2

approach incorporates annotations suggested by MIRIAM [Le Novère et al., 2005], and additional meta-information. In addition the approach was prototypically implemented as part of the search engine of BioModels Database, a standard repository for computational models.[2] The introduced concept and implementation are the first application of Information Retrieval techniques on model search in computational systems biology. Using the example of BioModels Database, it is shown that the approach is feasible and extends the current possibilities to search for relevant models. The advantages of this approach over existing solutions is the incorporation of a rich set of meta-information, and providing the user with a relevance ranking of the models found for a query. Better search capabilities in model databases are expected to have a positive effect on the reuse of existing models.

4.1. Background

4.1.1. Importance of model exchange and reuse

The study of a complex biological system now frequently includes the use of modelling and simulation techniques, in order to help understanding the system of interest, and to provide suggestions for promising experimental procedures [Klipp et al., 2009]. The rising complexity of *modelled* systems (see Figure 4.1, number of encoded species and reactions in BioModels Database [Glont et al., 2017, Li et al., 2010]), and the fact that research activities overlap between different research groups demand for model reuse. Modellers do not want, or cannot build their models of biological systems from scratch, but, on the contrary, need to seek for existing bits and pieces to build their models on, especially when composing complex systems by combining smaller sub-models (see for example [Endler et al., 2009, Liebermeister, 2008, Schulz et al., 2012]).

Standard formats for model exchange and open model repositories are crucial tools to make existing models available and accessible to the community as it becomes impossible to actually be aware of all existing models, and research groups involved in the modelling of a system of interest [Myers et al., 2017]. Some standard formats developed for model representation are widely accepted. Examples include the *Systems Biology Markup Language* (SBML, [Finney et al., 2003, Hucka et al., 2019]), CellML [Cuellar et al., 2003, Schreiber et al., 2018], or BioPAX [Bader and

[2] This resource was hosted by the EBI, Hingston, Cambridge, UK. Sadly, it became unavailable when the EBI-RDF Platform was lunched.

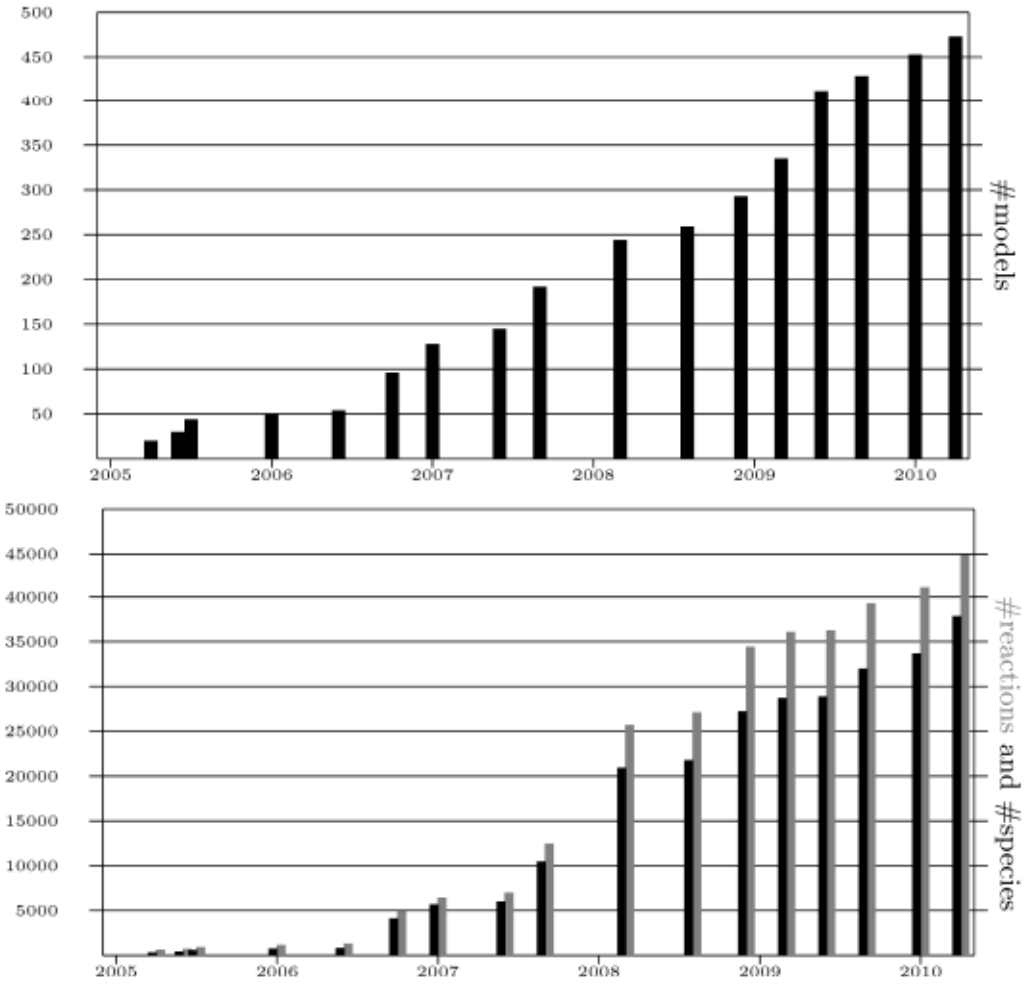


Figure 4.1.: *Growing number of Computational Biology models and their components in BioModels Database* Upper chart: Numbers of models in BioModels Database, as of January 2010. Lower chart: Number of species (black bar) and reactions (gray bar) in BioModels Database, as of January 2010. BioModels Database started with 20 models and a total of 322 species when it was launched in April 2005. In 2007 it already reached almost 200 models and 10482 species. The release in January 2010 recorded 453 models with 33702 species and 41069 reactions. Statistics for 2019 are presented in 8.1

Cary, 2005, Demir et al., 2010]. Computational models of biological systems (biomodels) in standardised representation formats are available from different model repositories, including BioModels Database [Chelliah et al., 2013, 2015, Glont et al., 2017, Le Novère et al., 2006, Li et al., 2010], the JWS Online Model Database [Lloyd et al., 2008, Sarwar et al., 2019, Yu et al., 2011, 2015], or the CellML Model Repository [Lloyd et al., 2008, Sarwar et al., 2019, Yu et al., 2011, 2015].

However, although getting more frequent, model reuse is not yet common-place. The reasons are similar to those hampering code reuse in computer science, where insufficient code documentation and missing modularization have been the biggest

hindrances [Tracz, 1988]. Most models are created using computational modelling environments; the constituents' names are often generated automatically and therefore are semantically poor. Models with unspecific species names such as $Po1$, $Po2$, $Pc1$, $Pc2$ (for instance, see model BIOMD0000000060 in BioModels Database), or unspecific reaction names $re1$ to $re76$ (model BIOMD0000000227 in BioModels Database) are common-place. A documentation of the names' meaning, amongst other things, is essential.

4.1.2. Standardised meta-information representation helps grasping models' nature

To countervail the problems experienced in computer science, efforts for the documentation of models' nature were developed. A minimum set of meta-information that is requested to be provided by many journals with each published bio-model is the *Minimum Information Required in the Annotation of a Model* (MIRIAM, [Le Novère et al., 2005]). Such meta-information provides a better understanding of a bio-model's complex and diverse *semantics* and, if computationally processed, enhances the model reuse.

MIRIAM meta-information encompasses general information about the model itself, e. g. the model's name, authors, or publication reference. But it also includes detailed descriptions of the model constituents, including the identification of encoded species, reactions, and compartments. MIRIAM itself is a textual recommendation, in form of a *Minimum Information* guideline following the MIBBI idea of coherent reporting guidelines for biological and biomedical investigations [Taylor et al., 2008].

A technical, standardised way of providing the MIRIAM-recommended meta-information is the *MIRIAM standard annotation* [Juty et al., 2012, Laibe and Le Novère, 2007, Le Novère et al., 2005]. The proposed format is a triplet referencing a piece of meta-information, also referred to as *annotation*, in an external resource. The reference to that meta-information is build of (1) the data type, (2) the identifier, and (3) a qualifier from a set of pre-defined qualifiers. Here the *data type* specifies the namespace within which to interpret the identifier. Some resources encode their knowledge as controlled vocabulary or ontologies. Among existing ontologies that are also used as data types by the MIRIAM standard are the *Systems Biology Ontology* (SBO, [Bernasconi and Masseroli, 2019, Le Novère et al., 2007]), the *Gene Ontology* (GO, [Ashburner et al., 2000, Consortium, 2018]), or the NCBI Taxonomy[3]

[3]NCBI Taxonomy: `http://www.ncbi.nlm.nih.gov/Taxonomy/`

[Federhen, 2011]. One advantage of using ontologies, i. e. "explicit specifications of a conceptualization" [Gruber et al., 1993], over free text information is the standardised encoding of biological knowledge that is then put into relation with other ontology terms. The MIRIAM standard *identifier* refers to the actual entry within the data type. It corresponds to the identifier (ID) the entry has in the external resource. Finally, the *qualifier* is used to characterise the relation between the annotated model element and the encoded meta-information. The possible qualifiers are defined at `BioModels.net` and include relationships such as `is`, `isVersionOf`, or `hasPart`.

For example, a `species` element encoded in a particular SBML model could stand for the compound "phosphosphoenolpyruvate" and in the model simply be called "PEP", offering little valuable information to the user. This compound, on the other hand, is described by the entry `CHEBI:18021` in the `Chemical Entities of Biological Interest` (ChEBI,[Degtyarenko et al., 2008, Hastings et al., 2015]) ontology. Referring to this particular identifier in that data resource by linking the resource and ID to the `species` element via the qualifier `is`, gives software and users access to a wealth of information independent of the elements name, such as synonyms, molecular and structural formulae and cross-links to other databases. Technically, the link is encoded in a standard form using URNs, e. g. `urn:miriam:obo.chebi:CHEBI%3A18021` for the given annotation. Another example is the annotation of a `reaction` element in an SBML document. Given a `reaction` element in a particular model stands for the "phosphorylation of glucose by hexokinase during glycolysis". This enzymatic reaction is also described by the `GeneOntology` entry `GO:0004396` (hexokinase activity). Attaching the URN `urn:miriam:obo.go:GO%3A0004396` to the reaction element using the qualifier `isVersionOf`, semantically enriches it and again gives access to further information, like alternative terms and enzyme nomenclature codes.

Extending the MIRIAM information

In order to enable a fine-grained retrieval of bio-models, [Köhn et al., 2009] proposes to consider even more information than MIRIAM's required one. Among them are versioning information on both the model and its annotations, but also information on the model encoding format, and information that is only related to the model, such as model behavior under certain conditions, simulation experiments applicable to the model, or simulation results available for the model. A detailed description of different kinds of meta-information considered in this work, even beyond MIRIAM is given in Knüpfer et al. [2006].

4.1.3. Finding models in model repositories using Information Retrieval techniques

As aforementioned, a crucial step for a computational system to return relevant models upon a user's query is the availability – and then incorporation – of meta-information on top of a model's structure [Köhn et al., 2009]. With the advent and growth of computational systems biology research, the number of bio-models available rapidly increases. For example, the number of bio-models available from BioModels Database is steadily growing, doubling about every 18 month[4] (see Figure 4.1, 8.1, number of models in BioModels Database). As a consequence, searching an existing model base for relevant models can result in a rather big number of models. Therefore, it is very important to support the user in *finding relevant* models in existing resources. It is common-place to leave the user with an unordered result set of models, without any explanation of why a particular model was found. For complex models the user is typically unable to grasp the model's nature at first sight [Köhn et al., 2009]. Having no information to assess *how good* a model matched his query, he cannot decide on its relevance. *Information Retrieval* techniques, which have been widely and successfully used in other areas, offer exactly these benefits for bio-model retrieval.

Information Retrieval is "the process to recover an information stored in a system (i. e. a database) on users demand" [Ferber, 2003]. One application for which the successful ranked retrieval of annotated documents has already been shown is *Multimedia Information Retrieval* (MIR). MIR models describe songs, images or videos annotated with different kinds of information, including meta-information like author or title, but also temporal or spectral information, as well as keywords. Currently, MIR distinguishes three independent classes of similarity measures depending on the kinds of identified features [Zhang et al., 2009]:

Metadata-based similarity measure (MBSM) defines queries by connecting keywords gained from the media object with Boolean operators like $\wedge, \vee$. Text retrieval techniques are then used to compare these query keywords with features of the multimedia objects.

Content-based similarity measure (CBSM) utilizes so-called low-level features, i e. automatically extractable items, such as rhythm. Queries make use of these features to search the content of music pieces. Different methods have been

[4] With the release of path2models [Büchel et al., 2013], over 180.000 automatically generated models were released.

developed to retrieve the items represented by low-level features, e. g. humming, tapping or query-by-example.

Semantic-description-based similarity measure (SDSM) evaluates meta-information on multimedia objects that are described with predefined words of different vocabularies.

Motivated by the above observations, a novel retrieval and ranking framework is described in the following. The framework takes into account different model meta-information to perform similarity-measure-based operations on bio-models.

Although *data* retrieval techniques have already successfully been applied to Life Science data in general [Esch et al., 2014, Lange et al., 2010], existing approaches do, however, not consider the retrieval and ranking of *models*.

4.2. Results and discussion

Subsequently an adapted version of the aforementioned solutions for MIR is applied on bio-model retrieval. To re-use MBSM for bio-model retrieval, the MIRIAM required meta-information on the model and its constituents is essential. Furthermore, usage of parts of the meta-information is suggested by Knüpfer et al. [2006] and Köhn et al. [2009]. When adapting CBSM techniques to bio-model retrieval, low level features (such as the encoded species, reactions, and so on) can be used. Finally, SDSM techniques can be used by tagging the models manually with relevant terms.

4.2.1. Definitions

A sophisticated retrieval and ranking approach necessitates a collection of k models from a pool of bio-models M and associated meta-information that is sufficient to rank the retrieved results with respect to a user's query. An annotated bio-model is defined as:

Definition 1 (Annotated bio-model) *An annotated bio-model $m \in M$ is described as a tuple $m = (m_S, m_A)$ of*

1. *model source code m_S in a machine-readable format*
2. *annotation information m_A describing the nature of a bio-model, and of its constituents.*

In the following, annotations of the model m are not distinguished from annotations of the model's constituents. All annotations will be processed equally, denoted as m_A. The annotation information m_A might be referred to as third party knowledge linked to m_S.

A feature is defined as:

Definition 2 (Feature) *A feature $f \in F$ is an attribute or aspect of a model m instantiated either through its model encoding m_S or its annotation information m_A.*

Definition 3 (Term) *Let T be a set of words called terms, then $\mathcal{P}(T) = \{\rho : \rho \subseteq T\}$ is the set of all subsets of T called power set.*

A model collection is then:

Definition 4 (Model collection) *A model collection C_M is a representation of M. Each $m_j \in M$ can be mapped on a $c_j \in C_M$ by splitting the model m_j into features $f \in F$ and their instances $\rho_f \in \mathcal{P}(T)$. So $c_j = \{(f_1, \rho_{f_1}), \ldots, (f_n, \rho_{f_n})\}$.*

Those definitions (1, 2, 3 ,4) hold for each model $m_j \in M$ classified into features and represented by $c_j \in C_M$.

A query is furthermore defined as (definition 5):

Definition 5 (Query) *A query $q = \{q_{f_1}, \ldots, q_{f_n}\} \in Q$ is a set of query parts $q_f \in F \times \mathcal{P}(T)$ with $q_f = (f, \rho_f)$; $f \in F$ and $\rho_f \in \mathcal{P}(T)$. All query parts q_f of a query q are pairwise disjoint.*

$q \in Q$ represents the user query. The parts q_f of q can either be mapped on the full set of defined features F, or on a subset of F.

Assuming a collection C_M of processed models M and extracted model features $f_i, \ldots, f_n \in F$, bio-model retrieval can be defined as follows:

Definition 6 (Bio-model retrieval) *An Information Retrieval model based on Baeza-Yates and Ribeiro-Neto [1999] is a quadruple (C_M, Q, FW, $R(q, c)$) where*

1. *C_M is a feature-classified representation of M*
2. *Q is a set of queries q, where each part $q_{f \in F} \in q$ can be mapped on a $f \in F$*
3. *FW is a framework for model representations, queries and their relationships*
4. *$R(q, c)$ is a set of ranking functions defining an order among $c \in C_M$ with regard to q.*

The framework FW realises the retrieval functionality. Each ranking function r, when applied to a query q, returns a ranked list of model representations c. The order of retrieved results is determined by the ranking function itself, the underlying collection and by the particular query. From the ranked list of feature-based model representations c_j, the ranking of the corresponding models m_j is deduced and represented in c_j.

4.2.2. Conceptual architecture of the framework

To perform ranked retrieval of annotated bio-models, a combination of text retrieval, ontologies, simulation dependent data, and model meta-data is used. The conceptual architecture for the developed retrieval and ranking framework is shown in Figure 4.2.

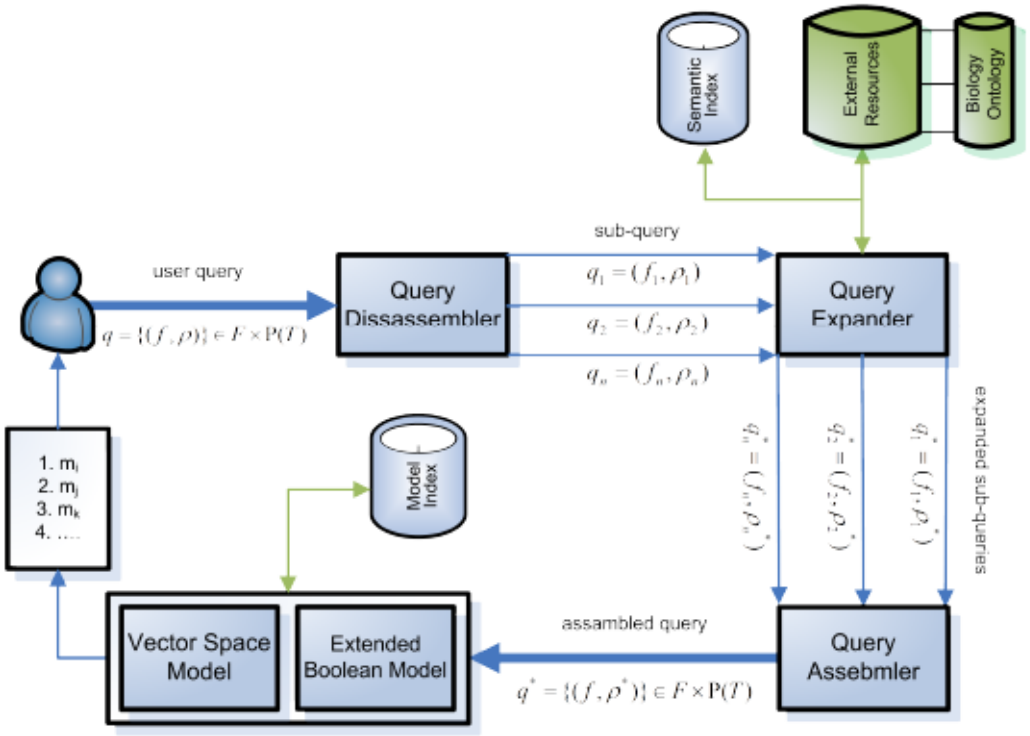


Figure 4.2.: *Conceptual retrieval and ranking architecture.* Overview of the conceptual architecture of the proposed ranking- and retrieval system. A version has been implemented in BioModels Database. The architecture shows the process of transforming a *user given query* by creating sub-queries, which are then assembled by enrichment of structural information and semantic indexing (see also Figure 4.4). The re-assembled query is then sent to the retrieval and ranking module, which makes use of the Extended Boolean Model to retrieve a list of matching models, and the Vector Space Model to rank the list of retrieved models. To determine the ranking, different weight information is used. Those are, however not shown in the given Figure.

For a user-given query q, consisting of a set of feature-assigned terms (f, ρ), a ranked list of models is returned. The ordered list of models $m_j \dots m_k$ is inferred from the order that is defined by the ranking function $r(c_j, q) > \dots > r(c_k, q)$, where c_j is the most relevant model representation with regard to the query q (see definition 6). To achieve this order, each query q is first disassembled into a set of sub-queries q_1 to q_n. Each sub-query q_i now contains a set of terms that will be mapped on a particular feature f_i. However, the query parts q_i are not directly executed on the data resources, but rather expanded using the QUERY EXPANDER. So far two different kinds of sub-queries are distinguished:

Semantic sub-query is any query addressing model constituents. This type of query is applied to the SEMANTIC INDEX.

Ontology sub-query is any query enriching the user query by finding related ontological terms. This type of query is applied to the BIOLOGY ONTOLOGIES.

All expanded sub-queries are assembled into a final query $q*$ which is sent to the retrieval and ranking system. The Extended Boolean Model [Baeza-Yates and Ribeiro-Neto, 1999] is used to select all models that are relevant to the query, and then the Vector Space Model [Salton et al., 1975] is used to define the ranking on those models. Both IR models work on the MODEL INDEX which contains all models and their associated URIs. The result of the process is a ranked list of model IDs.

4.2.3. Architectural components of the framework

Types of user queries

Information from different resources are processed and stored, and subsequently mapped on the frameworks internal structures; i. e. full-text indexes and databases. As a result it becomes feasible to answer very specific queries. Two different types of queries are distinguished. A query may consist of a number of terms (*query by value*, *QBV*) or of a complete set of features representing a model (*query by model example*, *QBME*).

Query by value (QBV) Using QBV, the user query q consists of features and free-text terms (f, ρ). The user given features f are a subset of all available features F.

Query by model example (QBME) Using QBME, a model forms the basis of a search for similar results, i. e. the complete set of features F is aligned.

Questions a user might have in mind are "Which models describe calcium concentrations in pancreatic cells?" (QBV), or "Are there any models dealing with the effects of caffeine on blood pressure in humans?" (QBV). One could also easily imagine to search for a model that "is similar to model BIOMD0000000227" (QBME).

Model index: incorporating model meta-information

The MODEL INDEX contains references to all models $m_i \in M$, as well as encoded information about constituents and meta-information.

Relevant features representing a bio-model were defined and grouped into several content-related *dimensions* to facilitate the creation of the bio-model collection C_M. Each of those dimensions has a certain importance associated to it, i. e. a measure of how relevant the information it carries is (see Table 4.1). *(1) Model constituents* is an important dimension which contains several features describing a model's constituents, e. g. species or reactions. *(2)* Information about authors, encoders or submitters of a model are grouped into a *persons* dimension. *(3)* Publications or published abstracts are contained in the *publication* dimension. The *(4) user generated content* holds information like keywords or tags. To restrict search results time-wise a *(5) dates* dimension holds time information, for example submission or modification dates. Finally, the *(6) administrative data* dimension contains specific information about the model file or the representation format used to encode the model.

The concept of dimension is a rather general one. Each dimension can, however, be refined into *features* f. A full list of features that make up the model index for all aforementioned dimensions can be found in Table 4.2. For example, the dimension *model constituents* is split into several features, among them *species*, *compartment*, *reaction*. Limiting a query to certain model features allows a user to be more specific. For example, it is possible to restrict a query `caffeine` to the feature *species* – and to disregard a "tribute to caffeine for the writing" in the *publication* feature. The values for each defined feature can be automatically extracted from a bio-model m if m complies with the given model definition 1. The additional assignment of weights for each distinct feature helps to determine similarity values, as will be explained later.

dim	Part	Importance	Description
1	`Administrative data`	low	administrative data like id, file name, file version, encoding formalism
2	`Persons`	medium	covers the author, encoder and submitter
3	`Dates`	low	submission or modification date
4	`Publication`	high	main publication or description of the model
5	`Constituents`	very high	information about the model constituents
6	`User generated content`	very high	additional user-provided information, e. g. keywords

Table 4.1.: *Importance of different information dimensions.* Information dimensions sorted by relevance. The information that is relevant for the characterisation of a bio-model's ranking is grouped into six different dimensions (dim). Each dimension has a different influence on the ranking. The least important dimension is the *administrative data*, the most important dimensions are the one encoding information about the model *constituents* and the one created from *user generated contents*.

Semantic index: identifying biological entities

Bio-model entities can be described by annotation information m_A encoded in MIRIAM standard URIs and stored in the Model Index. When searching for a model, a user cannot be expected to know the URIs for each biological entity of interest. On the contrary, searches for a constituent or bio-model must be possible using *characterising terms*, i. e. keywords. Therefore, the URIs must be parsed and the extracted information processed. The textual representation of each known constituent found in the external resources is resolved from its URI, and then indexed. By making it available for searching, *keywords* describing a model constituent can be used to retrieve models. For example, when searching for models dealing with caffeine, one may type either `caffeine`, `1,3,7-trimethylpurine-2,6-dione`, or even $C_8H_{10}N_4O_2$.

To map the textual descriptions, and also synonyms of a term, on a set of URIs representing the best matches for a defining term, a so-called SEMANTIC INDEX is used (see Table 4.3 for the structure of the Semantic Index). This index contains all URIs found in the models included. It furthermore is build of a column for each

Dimension	Feature	W	Dimension	Feature	W
Constituents			User generated		
(description)	modelName	4	content	-	-
	species	3			
	compartment	3			
	reaction	3			
	parameter	1.5			
	event	1.5			
	function	1.5			
	modelDescription	0.5			
(URI)	modelURI	5			
	speciesURI	5			
	compartmentURI	5			
	reactionURI	5			
	parameterURI	3			
	eventURI	3			
	functionURI	3			
Persons	author	4	Dates	creationDate	1
	encoder	1		modificationDate	1
	submitter	1			
Publications	publicationURI	5	Administrative data	ID	1
	publicationText	2.5		additionalID	1
				path	1
				content	1

Table 4.2.: *Assigned feature weights (W) by dimension.* Feature weights for the different model dimensions. Each dimension is further separated into the features it covers. For each feature, a specific relevance value, i. e. weight, is given. For example, in the *Constituents* dimension, one important feature for the model description is the `modelName`. The different URIs (`modelURI`, `speciesURI`, `compartmentURI` and `reactionURI`) also play an important role in determining the ranking. A less influential feature is the `modelDescription`, as for example found in the SBML `<notes>` tag.

existing qualifier. Every model m that contains a particular URI is added to the set of model IDs in the relevant qualifier column. The semantic index therefore enables to link a URI, resolved from search terms, to a set of bio-models within the collection C_M.

Having build the Semantic Index, queries may now be limited to models that use a particular qualifier to link a constituent to an annotation. For example, a user searching for `caffeine` can limit the result to models qualifying the annotation with `is` and `isHomolog`. The models using the query term in conjunction with `is` could

URI	qualifier			content
	bqbiol_is	bqbiol_isVersionOf	bqmodel_is	
urn:miriam:. obo.chebi. CHEBI:27732	BIOMD 0000000241	BIOMD0000000241		caffeine chebi 27732 chebi home advanced search browse ontology periodic ... moleculeschebimain caffeine chebi 116485 central nervous system stimulant caffeine ryanodine receptor modulator mutagen 1,3,7-trimethyl-3,7 dihydro-1h-purine-2,6 iuphar 1,3,7-trimethylxanthine dion msdchem d00528 kegg drug [..]
urn:miriam: kegg. compound: C07481	BIOMD 0000000241		BIOMD 0000000241	kegg compound c07481 entry c07481 compound name caffeine 1,3,7-trimethylxanthine formula c8h10n4o2 mass 194.0804 structure remark d00528 comment source coffea arabica tax 13443 xanthines reaction r07920 r07921 27732 knapsack c00001492 [..]
urn:miriam: kegg. compound: C00385	BIOMD 0000000015			kegg compound c00385 name xanthine formula c5h4n4o2 mass 152.0334 ko00230 purine metabolism caffeine metabolism [...]
urn:miriam: kegg. compound: C00048		BIOMD0000000221 BIOMD0000000222 BIOMD0000000219 BIOMD0000000218		kegg compound c00048 entry c00048 glyoxylate glyoxylic acid formula c2h2o3 mass 74.0004 structure reaction r00013 r00364 purine metabolism path ko00232 caffeine metabolism glycine serine null_1 threonine metabolism [...]
...				...

Table 4.3.: *Semantic index.* The semantic index is used to connect each existing URI in the database to the models in which it occurs. Thus, each column contains a set of IDs identifying a bio-model in C_M. Additionally it is stored *how* the URI is connected to the annotated model constituent (through the *qualifier* column). For each URI, the content, i. e. textual representation, that had been extracted from the ontology term corresponding to the URI, is normalised and indexed as well. A query can then be enriched by further related URIs (see also Figure 4.2, Ontology Query), resulting in an expanded query.

be ranked higher. This procedure also allows for weighting URIs differently according to their associated qualifiers.

The result of a query on the Semantic Index is a weighted, ranked list of URIs for each query term. That list is passed on to the Model Index where it represents a sub-query result that together with other sub-query results is assembled into a similarity value.

Biology ontology: incorporating similar constituents

Sometimes it might be useful to also include models with constituents that are *similar*, though not identical, to the one described by the original search terms, for example, if a search resulted in only a few models containing a particular constituent. BIOLOGY ONTOLOGIES expand a query by deriving similar constituents. A user searching for models encoding the constituent `caffeine` may also be interested in models containing the constituent `xanthine`, which is structurally related to caffeine.

To compare the relevance of a search term with terms in a particular ontology, a solution is proposed by Schulz, Liebermeister (discussed in personal communication[5]), who suggest to map different ontology Web resources on one common ontology. Using that ontology, the similarities of ontology terms are measured. The approach also takes into account different relations between the terms. In this, that approach is used to compute weights for ontology entries within a certain range of a given term. Apart from that method, other works from IR research exist which might be incorporated in later studies, e. g. [Li et al., 2003].

Incorporating weights

After retrieval, the relevant bio-models are ranked. The ranking function comprises weights derived from different sources. (1) The MODEL INDEX itself is used to incorporate weights derived from IR techniques such as term frequency – inverse document frequency [Baeza-Yates and Ribeiro-Neto, 1999]. (2) The importance of each feature is expressed by its weight (see Table 4.2). (3) A user may in addition assign a weight to a term in the query in order to increase that term's importance. (4) Preliminary results of the single sub-queries assigned to particular data resources are evaluated. (5) Weights derived from ontologies (see BIOLOGY ONTOLOGIES) may change the result ranking, e. g. models retrieved by ontologically derived terms can be ranked lower than others.

Ranking the results

All weights assigned to a model are used to determine the model's position in the vector space that is spanned by the Vector Space Model. Having all model positions identified the similarity can then be computed and the ranking inferred, based on the models' positions.

[5]Later published in [Schulz et al., 2011]

Qualifier	Weight
is	2.0
isHomologTo	1.7
hasPart	1.5
isPartOf	1.5
isVersionOf	1.5
hasVersion	1.5
isEncodedBy	1.3
isDerivedFrom	1.3
encodes	1.3
isDescribedBy	1.0
occursIn	1.0
hasProperty	1.0
isPropertyOf	1.0

Table 4.4.: *Qualifiers and their assigned importance.* The table shows the different qualifiers available from MIRIAM resources. The qualifiers are used in the SEMANTIC INDEX. Each qualifier has a particular weight assigned to it which reflects the strength of connection between a URI and a constituent.

4.2.4. Implementation: enabling model retrieval in BioModels

The introduced implementation is based on prior work on a general framework for testing different ranking functions on a given model base, called Sombi [Waltemath et al., 2011d].

Here an implementation for BioModels Database is presented. A necessary assumption is that the model source code m_S is provided in the open, standardised model representation format SBML. Furthermore, annotations m_A should be encoded using the MIRIAM standard annotation, i. e. MIRIAM URIs. The implementation[6] is based on the architecture presented in the previous section.

The advantage of using BioModels Database as a proof of concept lies in the amount of stored models – currently 241 curated, i. e. verified, models and additional 213 non-curated models (as of 2010-04-01). All models are encoded in SBML. All models in the curated branch are annotated, and as a consequence provide sufficient meta-information for a thorough testing of the ranking and retrieval system. Furthermore, analysing the stored information together with the BioModels.net team led to tentative weights for the different features (see Table 4.2), and helped on pinpointing the importance of different qualifiers (shown in Table 4.4

[6] All source code was freely available at the Biomodels.net SVN Sourceforge repository (`https://biomodels.svn.sourceforge.net`). The retrieval and ranking system was available online at `http://www.ebi.ac.uk/biomodels-demo/`. Sadly, both links are outdated.

The current BioModels Database search engine is extended and enhanced by including a greater number of features in the search process, by weighting different information, and by ranking the results according to the user query. Both types of queries, QBV and QBME are supported. The MODEL INDEX contains 454 models with 140977 terms separated into 25 features. The SEMANTIC INDEX contains 2261 URIs with 409124 terms. The used BIOLOGY ONTOLOGIES are NCBI Taxonomy, GO, ChEBI, KEGG Compound and KEGG Reaction [Ogata et al., 1999] (as of 2010-04-14). Candidates to include more formal (biological) semantics and information relevant to preserve a bio-model's semantics have been suggested in [Knüpfer et al., 2006].

The *Lucene Framework* [Gospodnetic and Hatcher, 2005] is integrated in the search system to create, maintain and search both the Model and Semantic Index. It provides retrieval functionality based on the Extended Boolean Model; its ranking possibilities are based on the Vector Space Model. To implement the retrieval and ranking process described above, Lucene has been extended and adapted, e. g. to enable different indices and sources such as the Semantic Index. While the implementation makes use of an adapted Lucene built-in similarity function, it will be useful in the future to provide advanced users of the ranking system with a collection of different similarity functions to choose from[7].

Search engine possibilities

Query by value Query by value allows the user to either perform a free text search querying all features, or a more sophisticated search selecting features of the different dimensions to be searched (refer to Tables 4.1 and 4.2). For instance a user is able to search for models having a certain author or for models including a particular "species". Furthermore, it allows to weight the different parts of a user's query using the specific *feature matrix* shown in Table 4.4.

Depending on the dimension selected, the query might be enriched or limited. This is especially important for the constituent dimension. For example, different terms describing a model constituent are used to query the SEMANTIC INDEX. The result is a list of weighted URIs, which is then used to identify a model in C_M in case the model itself does not provide the search terms the user queried. When searching a model by URI, the importance of an URI within the model is reflected through a qualifier; i. e. models encoding a URI with the qualifier `is` are more important than

[7] sota:sec:RankAggregation

models encoding the same URI with the qualifier `isVersionOf`. The weighting is done using the qualifier matrix shown in Table 4.4.

Additionally, the user is able to vary the importance of his search terms; i. e. one term describing a constituent can be more important than another. This *weight* is taken into account when computing the ranking. Besides the sophisticated ranking and retrieval system, the search engine supports common IR techniques like fuzzy search, range or proximity search, as well as wild-cards or phrase search [Baeza-Yates and Ribeiro-Neto, 1999].

Query by model example When querying by model example, the model used as a bait is analysed, and the values of extracted features are queried against the bio-model collection C_M. A ranked list of best matching models is retrieved. Enriched queries are switched off, as the example model itself provides sufficient contextual information.

4.2.5. An example for model retrieval and ranking

The following example illustrates the functioning of the reference implementation. If one wants to search for *recent models by non-bogus authors describing the effect of caffeine in human's digestive tract when drinking a cup of coffee*, the characteristics fulfilled by the resulting models are:

1. the model *should have* the compartment `gut` encoded
2. at least one species *must be* exactly `caffeine` (qualified using `is`)
3. the model *should* have been submitted later than 2008
4. the author of the reference publication *must not* be `John Doe`

That query can be submitted easily through the proposed advanced search interface of BioModels Database. The query is shown in Figure 4.3. The specification of different levels of requirements (should, must, must not) helps to be more specific in restricting the search.

To answer the query, the system first resolves the constituent `caffeine` into a set of URIs (SEMANTIC INDEX). Since the search for `caffeine` is restricted to the qualifier `is` (*must be exactly* `caffeine`), only the retrieved URIs that are linked to a model using the `is` qualifier are kept. Of those, a weighted list of URIs is build and then used for the feature `speciesURI` to query the MODEL INDEX. For

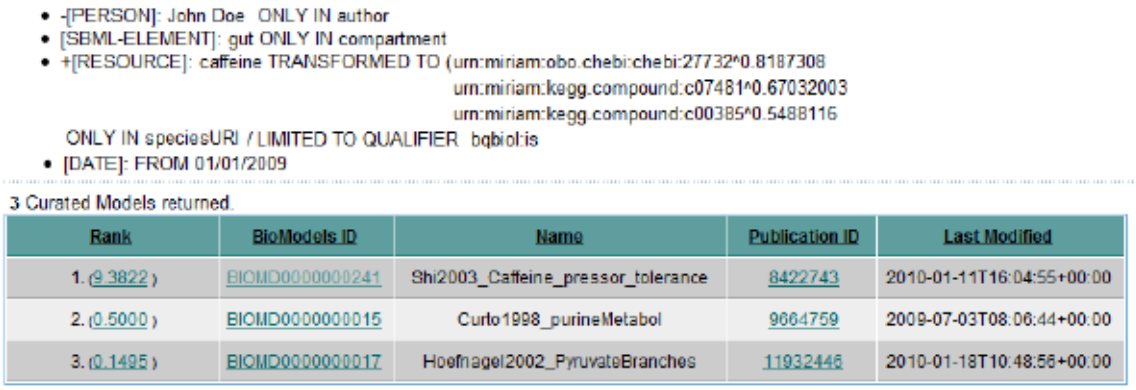


Figure 4.3.: *Sample query on the new BioModels Database search interface.* Screenshot of a part of the new search interface of BioModels Database. The interface allows to search for *Persons*, *SBML elements*, *Resources*, and allows to restrict the search terms to particular features using a single qualifier. Models may only be considered for a certain range of *dates*. The sample search correspond to *recent models by non-bogus authors describing the effect of caffeine in human's digestive tract when drinking coffee.*

the aforementioned example, the three best matching URIs are (a) `urn:miriam:obo.chebi:CHEBI%3A27732`, (b) `urn:miriam:kegg.compound:C07481` and (c) `urn:miriam:kegg.compound:C00385`. The URIs (a) and (b) both define caffeine, one in ChEBI [Degtyarenko et al., 2008] and one in KEGG [Ogata et al., 1999]. The URI (c) describes xanthine, a chemical structurally related to caffeine.

Together with the queries for `gut` in the component feature and *not* `John Doe` in the author feature, the MODEL INDEX query is internally assembled to:

```
+speciesURI:( urn:miriam:obo.chebi:chebi%3A27732 ^0.82
              urn:miriam:kegg.compound:C07481    ^0.67
              urn:miriam:kegg.compound:C00385    ^0.55)
 compartment:(gut)
-author:(John Doe)
 date:([01/01/2009 - *])
```

The prefix + and − denotes if a feature *must* or *must not* occur, no prefix implies the feature *should* occur. The ^ denotes the weight assigned to the sub-query results retrieved from the semantic index.

The Extended Boolean Model is applied to query the index for each feature independently (speciesURI, compartment, date and author). The preliminary results are four sets of matching internal model identifiers. These sets are then conjuncted using Boolean algebra and taking into account whether a feature *should*, *must* or *must not* occur.

4. Model Ranked Retrieval

In a second step, the results are ranked using the Vector Space Model, according to the different types of weights. The predefined feature weights (Table 4.2) put a particular importance on the speciesURI feature. Thus, all models that matched the speciesURI feature are ranked high, incorporating the weight created by the sub-query to the semantic index. If a retrieved model, besides the mandatory features (*must*), matches additional optional features (*should*), the scores are summed up, resulting in a higher rank. In this case, the feature "date" is not very important – thus, it results only in a small increase of a model's score if the feature matched. The ranked results for the sample query performed on BioModels Database is shown on Figure 4.4.

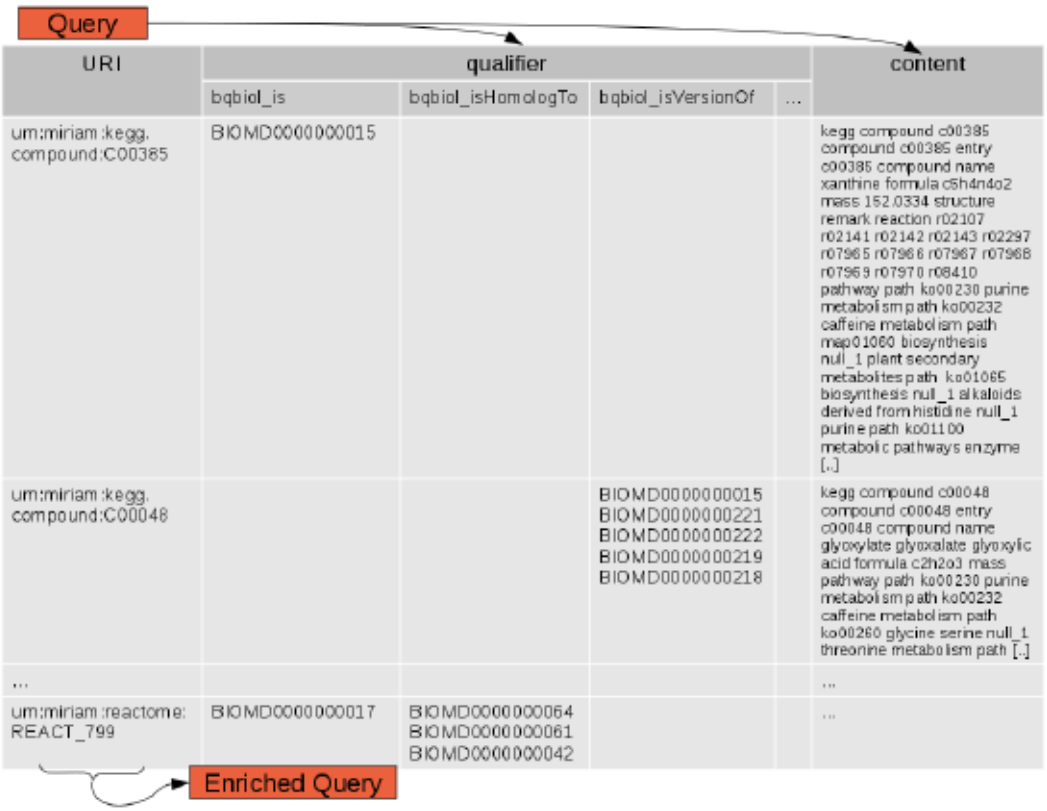


Figure 4.4.: *Ranked results.* Search result obtained on BioModels Database with the given sample query (see Figure 3). The upper panel shows the enriched query. Due to the precise formulation of the query, and the requirement that caffeine must occur and additionally must be qualified with `is`, the result contains only three hits. (1) This model matches the top two constituents resolved by the `semantic index`, and additionally the term *gut* in the compartment feature. (2) The model matches the constituent ranked third by the semantic index. (3) The lowest ranked model only matches one constituent ranked eight by the semantic index - this is a very weak relation resulting in a very low rank.

4.3. Conclusions

The framework presented here describes, to the best knowledge and for the first time, the application of Information Retrieval techniques on Computational Biology models. The theoretical method relies on knowledge extracted from model annotations, but also incorporates context information. The BioModels Database implementation presents a practical example of this method. It enhances significantly the search possibilities of BioModels Database users. Thorough evaluation, for instance using F-measures, is needed, but currently difficult due to the lack of reference to compare with. The concepts' generality ensures it is easy to apply to other models bases.

There's always going to be a separate version of you that people will create, and you have no control over it.

— *Rebecca Hall*

5. An inhabitants life cycle

Improving the reuse of computational models through version control [Waltemath et al., 2013a] describes prerequisites to use algorithms for model provenance management. It was the first systematic approach to model provenance in the domain of systems biology and also offered a prototypic implementation for difference calculations of SBML (later also CellML) models[1].

Research question: Only models that are accessible to researchers can be reused. As computational models evolve over time, a number of different but related versions of a model exist. Consequently, tools are required to manage not only well-curated models but also their associated versions.

Results: In this Chapter conceptual requirements for model version control are presented and discussed. Focusing on XML formats such as SBML and CellML, methods for the identification and explanation of differences, and for the justification of changes between model versions are elucidated. In consequence, researchers can reflect upon these changes, which in turn has considerable value for the development of new models. The implementation of model version control will therefore foster the exploration of published models and increase their reusability. In summary, this chapter explains why model version control is an integral part of model management.

5.1. Introduction

Modeling has become an integral tool for research in computational biology [Finkelstein et al., 2004, Wolkenhauer, 2014]. The increasing impact of modeling for biology

[1] see Appendix A.2

is reflected in the rapidly growing number and complexity of computational models of biological systems (in the following called "models") [Henkel et al., 2010, 2015, Li et al., 2010]. Current modeling projects such as the *Virtual Physiological Human*[2] require the usage of techniques for model coupling, merging, and combination at different scales. Computational support is needed to manage models, to ensure model exchangeability, stability and result validity, and to foster communication between project partners.

Model representation formats standardize model encoding. Examples are the *Systems Biology Markup Language* (SBML) [Finney et al., 2003, Hucka et al., 2003, 2019], *CellML* [Cuellar et al., 2003, Schreiber et al., 2018], or *NeuroML* [Cannon et al., 2014, Gleeson et al., 2010]. These formats represent a model's structure (e. g. the biochemical network) and allow basic annotation of the model to better convey a model's intention. For example, SBML developed an annotation scheme [Hucka et al., 2010] which reuses the *Resource Description Format* (RDF) [Lassila et al., 1998a] and identifiers from the MIRIAM Registry [Juty et al., 2012, Wimalaratne et al., 2015]. The valid description of a model is a requisite for its dissemination, but an additional descriptive layer is necessary to ensure direct result reproducibility. This layer is covered by the *Simulation Experiment Description Markup Language* (SED-ML) [Waltemath et al., 2011b] which is a format for the standardized encoding of simulation experiment setups. Further projects develop standard formats for result data (e. g. SBRML[3]) [Dada et al., 2010] or graphical representations of models (SBGN) [Le Novère et al., 2009, Touré et al., 2018, Van Iersel et al., 2012]. With an infrastructure at hand that provides modelers in computational biology with a rich set of model-related information it is time to think about integrated management solutions for models, their associated simulation experiments, result data, reference publications, etc. [Henkel et al., 2012b, 2015].

Curated model source code is published in model repositories, for example *BioModels Database* [Chelliah et al., 2013, 2015, Glont et al., 2017, Le Novère et al., 2006, Li et al., 2010], the *Physiome Model Repository* (PMR2)[Lloyd et al., 2008, Sarwar et al., 2019, Yu et al., 2011, 2015], *ModelDB* [Hines et al., 2004, McDougal et al., 2017, Peterson et al., 1996], or the *Open Source Brain* [Gleeson et al., 2019]. Open model repositories grant researchers access to published models. They provide a platform for model sharing and long-term storage, and they add support for model validation, curation and annotation of submitted models. Several publishers already ask for

[2]Virtual Physiological Human: `http://www.vph-noe.eu/`
[3]SBRML `https://sbrml.sourceforge.net/SBRML/Welcome.html`

model source code to be made available along with the written paper (including Oxford journals, BMC journals, PLoS journals, PeerJ or the FEBS journal). They recommend upload of model code to open model repositories using standard formats and annotations. Consequently, open repositories are high-quality, reusable resources of models. However, one disadvantage of current systems is the unavailability of user-interpretable version information together with the subsequent lack of model histories.

Each model is subject to a characteristic set of changes that reflects the model's updates from its creation to curation, publication, and later reuse in other contexts (refer to model circle of life, Figure 1.2). One prerequisite for the study of a model's history is the availability of all significant model versions. Here the model version control system (VCS) saves time in recapitulating the different modeling steps taken to build a model. This aspect also becomes relevant when publishing the work; teaching model design in courses; or during model curation when curators need to discuss necessary model changes with authors before publication in a model repository. A VCS is capable of storing all existing versions of a model during its existence. Subsets of these versions can easily be filtered and displayed to the users. The level of detail depends on the specific application. For example, the developers of a model repository may choose to offer all versions that reproduce the results in the reference publication. However, during the collaborative development of a model, all intermediate versions may be of interest for the project members.

The need for model version control has been previously discussed in research groups facing model evolution in computational biology [Beard et al., 2009, Cuellar et al., 2002, Hucka et al., 2010, Li et al., 2010, Miller et al., 2011]. In general, VCSs such as Subversion[4] (SVN) or Mercurial[5] track every change made in a source document, along with information about who made the change and an optional log message containing information why a change has been made. PMR2 has recently demonstrated how Mercurial can be used to track and present model versions over the web [Miller et al., 2011]. Once processed, the information that is recorded by a VCS enables users to study a model's past and to answer specific questions about the model (Figure 5.1). Modelers may investigate which parts of a model have been changed, and how these changes were justified. For example, changes in the model parametrization may be justified by a new publication. It is also interesting for a modeler to see how often a model has been changed, and when it was last changed.

[4]Subversion: `http://subversion.apache.org/`
[5]Mercurial `http://mercurial.selenic.com/`

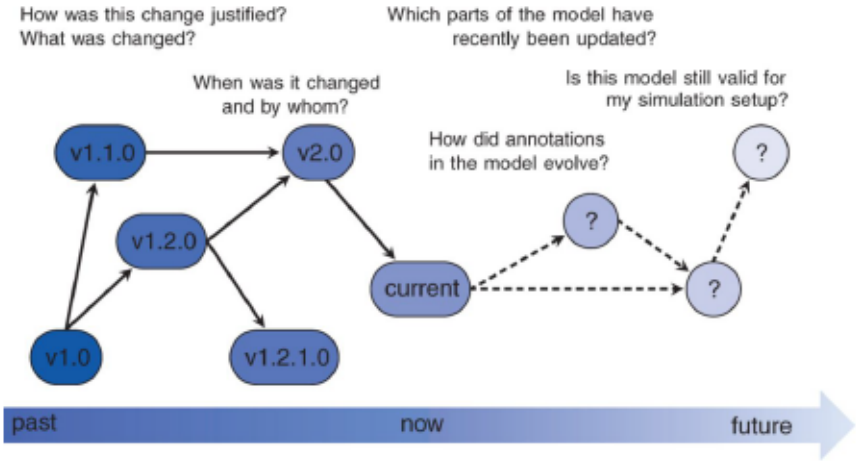


Figure 5.1.: *Questions that a system for model version control can answer.* Managing the versions of a model in a version control system is beneficial. Such a system tracks all relevant changes (What has changed? Why?). Questions about the model's history can be answered. In addition, researchers can study the likelihood of future changes and have a measure for a model's quality (Which parts of the model have recently been updated? By whom?). When models are linked to simulation experiments, consistency checks can be performed based on the detected model changes.

This information indicates how recent a model is and whether it has been tested and used by the community. When reusing model code which has been changed frequently in the past, the modeler might decide to confirm the model's validity on a regular basis. Simulation experimental setups are valid for a model at a certain time (or range of time). Changes in the model may directly affect the simulation results and must therefore be communicated and explained. In addition, the applicability of standard simulation experimental procedures, termed *functional curation* [Cooper et al., 2011], must be ensured for each single version of a model. Consequently, the availability of every model version used in a simulation experiment is a major requirement to ensure reproducibility of results.

In summary, the awareness of a need for version control led to the incorporation of technical solutions for model version control in model repositories. However, current approaches are appropriate for software code rather than model files. The standard algorithms that are to date implemented perform poorly on XML-encoded models. Existing solutions also miss support in aggregating version information about a particular model into a model history and interpreting changes. Implementing tools that identify and highlight differences between models enable researchers to reflect upon model changes which in turn has considerable value for the development of new models. This Chapter discusses concepts for model version control and provides

solutions to the previously mentioned problems. The focus is on models in XML-based standard formats. The presentd methods are document-centric and therefore perform well on single-document models, e. g. SBML Level 2 models. However, the discussion section also elucidates on how multi-document models can be managed, e. g. CellML models with imported model components or SBML Level 3 model using specific packages. This Chapter describes, to the best knowledge, the first approach of a document- and entity-centric version control for computational biology models.

5.2. Results

A model VCS should be tailored to existing model representation formats, which are typically XML- and RDF-based. It should furthermore reflect the temporal evolution of a model and present model changes to the users. In the following, current approaches to model version control are reviewed with respect to these requirements. Afterwards, a number of concepts for improved model version control are presented.

5.2.1. Current approaches to model version control

Two major approaches to model version control are currently taken: Documenting changes directly in the model file *versus* addressing version control inside the respective model repositories. Both approaches are outlined in the following.

Version control inside model representation formats:

The different standardization communities offer means to store version information inside the respective model representation format. For example, the CellML metadata specification [Cuellar et al., 2002] distinctively stores "trivial changes" (e. g. error corrections during the translation of a model to CellML) and "substantial changes" (e. g. creating a new revision of a model) [Beard et al., 2009]. SBML implements the model history as an additional element of the language itself which can be attached to any SBML element. The main objectives for the SBML history element are the provision of information on the creators of the encoding and the provision of modification dates (recording the creation of the model constituents and their subsequent changes).

In general, the storage of version information inside the model representation ensures that all changes applied to a model are shipped together with the model code. However, the disadvantage of this approach is that the description of changes

inside the model file is in itself already a change of model code. Consequently, tools which are in principle capable of calculating differences between two files will incorrectly identify added history information as changes, even though the model code updates do not reflect changes in the model. Furthermore, the history concept in representation formats is rather limited and not fully supported by software tools. The quality of the history is highly dependent on manual additions provided by the model authors and therefore more often than not incomplete.

Version control inside model repositories:

As an alternative, model repositories may implement support for version control. PMR2, for example, builds on a distributed VCS [Miller et al., 2011] based on Mercurial and, nowadays, on Git [Yu et al., 2015]. Instead of hosting one central repository, each developer may keep a local instance. Changes can then be pulled from and pushed back to PMR2. Once submitted, models can subsequently be imported as a particular revision of a workspace. The workspace revisions also enable authors to review the change sets of models. Change sets are the collection of all differences between two versions of a model. CellML offers continuous version control at all times using exposures. An exposure (snapshot) is a publicly viewable presentation of a particular revision of a model. It may contain any type of generic text based files (e. g. experimental setups, documents, or simulation experiment descriptions). ModelDB also provides preliminary version support for a subset of the available models. Similarly to the CellML model repository, it uses an implementation of Git. Here, a collection of models includes all revisions and associated versions of all models, and a log is available. This log contains the age of each model version, the submitter and a short description. It can be explored in multiple ways, including a tag-based view, a graph-based view, or a branch-based view. BioModels Database utilized the common non-distributed VCS Subversion[6] . The original file (i. e. the first uploaded version of the model in its original format) is provided on the model overview page, but also the SBML model version at each release point is available. BioModels Database distributes new versions of a model approximately four times a year with each release of the database.

Recapitulating, an integrated history inside a model repository ensures the availability of all model versions at all times and the consistency of model files. However,

[6]To allow users to retrieve and compare different versions of a model and its annotations in the future, BioModels Database developers plan to extend the functionality of their version control system [Li et al., 2010]. According to Glont et al. [2017], BioModels still uses a snapshot based system to keep track of a model's development history.

only a complete checkout of a model, for example using PMR2, keeps the link to previous versions. Once a user downloads a model file from a repository he loses this link. Furthermore, in 2013 no software tool allowed displaying a model's history in a particularly useful format; neither can model histories be meaningfully interpreted.

5.2.2. Concepts for improved model version control

In the following, solutions to selected problems identified in the analysis of current systems for model version control in computational biology are presented. The proposed new approach includes XML-aware difference detection, change transparency, and justification of changes. These concepts will be explained in detail in the following paragraphs.

XML-aware difference detection:

VCSs that are used in software development are designed for programming code and text documents. Examples of such systems are Subversion, Git or Mercurial. They implement a line based algorithm that relies on the *Longest Common Subsequence* (LCS) [Hirschberg, 1977]. LCS identifies changes between two files by matching the longest common subsequence of characters in them. A major concern with existing approaches to model version control is the choice of algorithm for difference detection. The set of operations that describes the transformation of a file version into its successor is denoted as *diff*. A diff between two versions of a model must contain all changes in terms of operations and yet be complete, minimal and interpretable. At a first glance, model code is in the same way semi-structured as is source code; both contain structured and free-text elements. In fact, BioModels Database stored the different releases of model files in an Subversion [Li et al., 2010], and the CellML model repository implements a Mercurial based system [Miller et al., 2011], and a Git based system [Yu et al., 2015]. Both solutions are LCS-based. However, unproven is the applicability of several methods implementing the LCS algorithm on difference detection in models. Line based algorithms perform poorly in the difference detection of models. The main reason is that LCS does not respect the XML structure [Rönnau et al., 2005] that constitutes model files. The XML format may already change when a model is loaded and then exported by a simulation software, e. g. leading to changes in the indentation of XML code or to the reordering of XML elements. Model code is often automatically generated in software tools, sometimes without the awareness of the user (e. g. the SBML representation of a model in COPASI). Each software

tool has its own preferred way of representing the model code, sorting the occurring XML elements, or breaking lines in the XML code. Furthermore, people tend to open downloaded models in text editors which reformat the XML code automatically. These quite common changes are detected by the LCS-algorithm, but they are in fact irrelevant for the model's history and would be neglected by entity-based algorithms. In other words, while being successfully used for source code version control, LCS is not suitable for XML version control [Chawathe et al., 1996]. Moreover, Krause et al. [2010] already mentioned that the LCS-based systems do not offer sufficient branching and merging options for XML-encoded data; these features have to be implemented on top of existing tools.

Due to the format of existing model representations, the idea is to reuse and adapt algorithms for difference detection in XML documents. Some systems are available that specifically work for XML documents, e. g. [Chawathe et al., 1996, Rönnau et al., 2005, Rosado et al., 2009]. They build on standard XML diff algorithms such as *Diffxml*[7], *XyDiff*[8], or its Java-based variant *JXyDiff*. These algorithms can be used to detect differences in models. However, it should be noted that merging XML file versions, in general, is an ongoing research topic in information systems [Krause et al., 2009, Rönnau et al., 2009, Schulz et al., 2006]. The approachdescribed in the following facilitates the XyDiff algorithm to create a library for difference detection, *BiVeS* (see Section 3).

Change transparency:

Entity based and XML-aware algorithms reliably identify the differences between two versions of a model. This information must be stored in a way that can be interpreted and explored easily by researchers. For example, a model's parameters might have been updated due to new scientific findings, and it is then important to know which parameters were changed and what the old and new values are. However, these updates in model repositories are to date not propagated to the user. The argument here is, that there should be a way for modelers and curators to identify the particular versions of a model that exist. Moreover, users should be able to easily identify who contributed to a particular model version. Multi-partner projects can result in models which originate from several different partners, e. g. the metabolic yeast model [Herrgård et al., 2008]. In these cases, a mechanism for version control can identify and manage the single contributions of authors for each version of the

[7]Diffxml: `http://diffxml.sourceforge.net/`
[8]XyDiff: `http://leo.saclay.inria.fr/software/XyDiff/`

model.

Another reason to propagate model changes is a model's use in simulation experiments. A SED-ML description does not necessarily contain the models which were used in the experiment, but it often links to the models by an unambiguous identifier. For example, the reference to the Repressilator model in BioModels Database is `identifiers.org/biomodels.db/BIOMD0000000012`.
An update of that model in BioModels Database may lead to invalid or unexplainable simulation results when executing the SED-ML file, e. g. if an observed species was deleted or renamed in the model code. When executing a SED-ML file which reuses that model it is therefore crucial to alert users about the changes.

In summary, it is not only important to inform users on a model update *per se* but also to provide information about what has been updated. These requirements can be realized with the aforementioned diff files which store all detected differences between two model files. Here the XPath concept is used to unambiguously identify the location of each such difference. The diff file itself can again be encoded in XML. An alternative solution was suggested by Saffrey and Orton [2009] who proposed to store XML patches in the context of version control for SBML encoded pathway models. A patch contains information about how to convert the original model into its successive version. Saffrey and Orton [2009] proposes a converter while the main idea of the aproach gescribed here is an explicit listing of all changes in a model.

Justification of changes:

All relevant changes should be justified with references to a reliable source explaining reasons for that change. The understanding of changes leads to a better confidence in the model and thereby encourages researchers to reuse existing models[9]. For example, a model curator should be able to annotate the update of model parameters with the publication providing ground for the new values. If a model in a repository is updated due to new biological insights, an explanatory link to a public database can be provided. Also, an updated kinetic parameter could be justified with a link to the SABIO-RK database [Wittig et al., 2006] which contains information on curated reaction rates, parameters, and their experimental conditions.

A technical mean to encode such justifications are references to external resources such as controlled vocabularies or ontologies. MIRIAM annotations, which are already used for model annotation, could also be used for difference annotation.

[9]The COMODI Ontology was created to server this purpose. COMODI is described in Chapter8.2

Location		Type	Operation
XML	Annotation	mathematics	update
element	qualifier	biology	insert
attribute	URI	parameter	delete
value		typo	move
		...	

Table 5.1.: Controlled vocabulary for change classification.

Table 5.1 shows a possible coarse-grained classification of types of model changes to exemplify the use of annotations for the characterization of model changes. The draft version distinguishes the location of changes (XML *versus* annotation) from the type of change (typographic corrections *versus* changes in the underlying biology). All changes can be mapped on specific XML operations (update, insert, delete, move). Each difference between two versions of a model can be annotated with terms from this classification, and several annotations might apply to one change. For example, each change occurs at a particular position in the XML file and therefore can be marked with a term from the Location branch, i. e. `attribute`, `element` or `value` change. In addition, a change that is annotated with the terms `parameter` (from the Type branch) and `update` (from the Operation branch) represents an update in a parameter value of a modeled entity.

When implementing a history-aware software tool in a model database, the above classification can be used to display only relevant changes (with respect to a particular application), e. g. during curation. The classification can, in addition, be used for automated reasoning. For example, a new version of a model that only contains annotation updates does not need to undergo a validity check on associated simulation experimental setups, as the model structure is not affected by the annotation updates. To implement reasoning on model changes, reusing existing methods for reasoning on SBML models, such as the ones developed by Hoehndorf et al. [2011] is promising.

5.3. Implementation

The software library *Biochemical Model Version Control System* (*BiVeS*) implements the above-mentioned requirements. The focus is on models encoded in SBML format, however *BiVeS*is capable of parsing CellML [Scharm et al., 2015]. The methods are similarly applicable to other XML-based model representation formats such as NeuroML, and comparable. To use *BiVeS* with other model representation formats, the model parser needs to be extended. The restriction on XML-based formats allows

to reuse algorithms for XML version control. *BiVeS*is based on the existing XyDiff algorithm.

The focus on SBML is rooted in the high number of models available from BioModels Database for testing. First, all existing versions of the curated models that were available from BioModels Database releases[10] were downloaded. Second, *BiVeS* were tasked to find changes between the single SBML model files [11].

5.4. Discussion

All relevant changes applied to a model should be tracked and they should be fully listed and documented. Users need to know what has been changed in a model, by whom, why, and when [Beard et al., 2009, Scharm et al., 2016]. Surprisingly, these requirements are not yet covered by existing systems.

Improving model version control:

To date, model files are rarely accessible by the users of model repositories in formats other than the original model file, or the processed, curated model file. For example, SBML files from BioModels Database are available for every release point. However, model code might change between two releases. These changes are not traceable by the community. Furthermore, the current BioModels Database releases are snapshots of the repository; they do not relate different versions of a model. Also the concept of a model history inside SBML is not intuitive; different steps must be taken to reconstruct the history of a model in terms of curation and modification dates. The types of changes are not encoded in SBML models. The model database PMR2 already provides a system for model version control. The repository stores exposures and Mercurial logs which represent different versions of a model. However, the interpretation of changes remains difficult, and the system lacks a visual representation of the model history.

A consolidated view of all these factors indicates that a sophisticated model VCS is missing in current software tools. The system proposed here identifies and classifies the changes in a model. Instead of using built-in line based difference detection (as used by SVN) it provides the XML aware difference detector *BiVeS*. Figure 5.2 shows

[10]The release are no longer available as BioModels switched to a rolling release schema

[11]Examples for models in different versions and their calculated and visualized differences were available on a testing platform called *BudHat* at `http://sems.uni-rostock.de/budhat`. Sadly, this link is not available any more as the SEMS project is terminated.

how a model history can be maintained, and differences between two model versions can be detected and then visualized in *BudHat*. Such a graphical representation allows users to quickly grasp the difference between two model versions and to understand the evolution of a model over time. However, a number of open questions

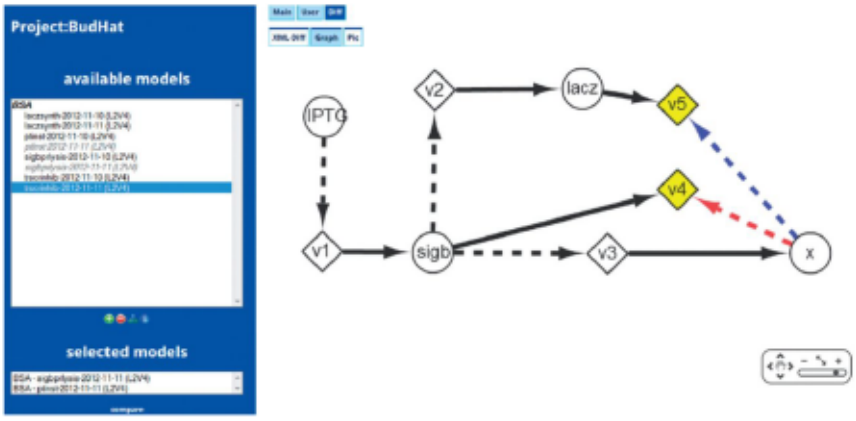


Figure 5.2.: *The BudHat tool for model version control.* The figure shows the prototype implementation of *BudHat*. Users can login to create their own history of model files (private or shared). The list of available models is shown on the left hand side of the figure. Two models from that list can be chosen and compared. The graphical representation shows the differences between these two model files. The differences are calculated by the *BiVeS* library. The result can be viewed either as an XML description of changes, or be displayed graphically, using an adaptation of the Cytoscape web tool. All changes in the network structure are color-coded as follows: new elements in blue, deleted elements in red, updated elements in yellow. A second feature is the graphical representation of a model's history (not shown in this figure). It provides a quick overview of existing versions of a model in *BudHat*.

remain for future investigations despite the improvements introduced here.

Leaving the choice to the user:

Providing a user with the differences between model versions enables him to actively decide whether or not to use an updated version of a model. For example, upon discovering that a model has solely been updated to a new version of SBML, a user may decide to keep the older version of the model until his software is updated to support the new format specification. In this case the model remains biologically valid. A decision for a model version requires a user to be aware of the model's change compared with its preceding version. This decision is supported by a visual representation of changes, prototypically implemented in *BudHat* (see again Figure 5.2). Ongoing research in the field of data and network visualization can

contribute to this problem.

Models must be unambiguously identifiable:

Reproducibility of results, based on a simulation setup, requires the simulated models to remain accessible. SED-ML files use the identification scheme provided by the MIRIAM registry[12] to link to the SBML models involved in a simulation. However, it is currently impossible to address a particular version of a model. Only the latest version of a model is addressable. For example, the identifier for the Repressilator model in BioModels Database is `identifiers.org/biomodels.db/BIOMD0000000012`. The identifier is the only entry point to a set of versions of the SBML model, and no mechanism exists to reference a particular one of these. This scheme potentially leads to errors in the related simulation descriptions as the person executing a SED-ML file cannot know whether the model has been updated since its last execution. Sometimes a software may not even be able to run an updated version of a model. Consequently, a scheme for the unambiguous identification of model versions is required. It should be noted that the developers of PMR2 already demonstrated how unambiguous links for each revision (exposure) of a model allow to reference single versions of a model [Miller et al., 2011]. This concept could provide a valid solution for other model repositories, too.

Version control for multi-document models:

This approach to model version control described here is document-centric, meaning it handles only single-document models. However, with the development packages in level three of SBML, different kinds of multi-document models can be generated. Also in CellML, the concept of importing model components results in multi-document models. As *BiVeS* is document-centric it cannot *per se* handle multi-document models. Two different solutions can be outlined: *BiVeS* could be used as a stand-alone tool to detect the differences between model versions. In this case, the multi-document models must be flattened (i. e. merged into one single file). A second solution is to combine the VCS with a database in the background handling multi-document models. Consequently, the problem is shifted to the storage layer of the model management system. The creation and usage of such a model menagement system is shown in Chapter 6. It describes how SBML and CellML models can be transformed into a graph-based representation and then be managed by a graph database [Henkel

[12]MIRIAM registry is now incorporated into `Identifiers.org`: `https://identifiers.org/`

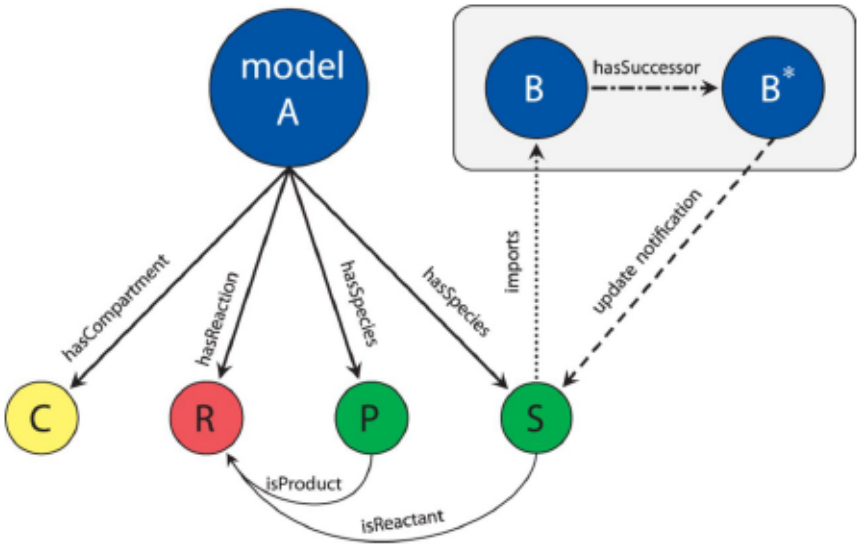


Figure 5.3.: *Version control for multi-document models.* The figure shows an example for multi-document models: Here models A and B are stored, and model B has a successor B*. Furthermore, component S in model A imports model B. When managed in a repository, the information about occurring changes in model B can also be propagated to model A which is using model B. More precisely, when version B* is created, a notification link to model A's component S can be created in the database. Thereby, the repository marks component S as changed, resulting in further updates if model A itself has been used as a component of another model. As is not possible for now to rule if changing model B to B* has an impact on model A, it is up to the user to resolve possible conflicts.

et al., 2012b, 2015].

This approach can be extended to enable version control for multi-document models (Figure 5.3): Using a graph structure allows for relating models and model constituents. An import relation in the database links the model to constituents that are defined outside the model file, thereby representing multi-document models. Subsequent changes in one model can be propagated to all models reusing the updated components as a constituent. Figure 5.3 shows an example of a multi-document model and outlines how the relations between database nodes can be used for checks on updates in single models.

Version control for systems of models:

All ideas in this Chapter discussed the identification of changes between two versions of a single model, and the previous section outlined solutions to handle models that are composed of more than one document. However, even further information can be gained when looking at the evolution of models for a particular biological system.

For example, a significant number of models that are based on the early encoding of Tyson's cell cycle model [Tyson, 1991] have been submitted to and published in model repositories. Looking at the different cell cycle models, how they relate to each other, how the single models evolved, and how parts of models were merged into new models requires not only a version control of single XML files but also of a collection of models. One approach for addressing the problem of linking entities across model files is the use of annotations, for example using the *Computational Neuroscience Ontology* (CNO[13]) which contains a branch on model classification. However, the definition of similarity measures for changes across models is still an open research question.

What is a new model – and what is not?

Models may exist in different versions over time and they are generated by different research groups. Furthermore, researchers may produce a number of instances of a model with differing parameterizations [Finkelstein et al., 2004]. The fact is that models must be accessible for reuse and to foster result reproducibility. Both reduce costs and effort during model development. Consequently, questions such as how to distinguish model versions from each other and how to separate models from simulation setups to avoid the unnecessary redefinition of models as "new models" must be discussed in the future.

5.5. Conclusion

Until 2015 technical solutions to the problem of model version control did not allow users to access, study, compare, and visualize different versions of a model [Scharm et al., 2015]. As of today, an implementation based on the ideas described in this Chapter exists and is described in more detail in Chapter 8.2. To achieve such an implementation the following requirements are summarized here:

1. **An XML-aware algorithm for difference detection should be used**: Moving from LCS to entity based algorithms for XML version control is a first step towards a more efficient model version control system. Entity based change detection is minimal, as it neglects XML formatting changes. Consequently, the extension of XML-aware algorithms for difference detection in models should be the preferred solution.

[13]Computational Neuroscience Ontology: `http://purl.bioontology.org/ontology/CNO`

2. **All changes should be transparent to the user**: It is essential to know *which* parts of a model have been changed, and *how*. This information can be encoded in a list of differences for pairs of two model versions. Consequently, users should be provided with a visual representation of changes between model versions.
3. **Justification should be given for each change:** It is important to provide justification for each relevant change in a model file, thereby encoding *why* something was changed and *by whom*. Entity-based difference detection enables partially automatized annotation of changes and their classification. Therefor a desire is to establish tools for the semi-automatic annotation of model changes with terms from a controlled vocabulary.

These requirements should be respected in software tools that offer model management. *BiVeS*, a library for model change detection that can be integrated with existing databases or version control systems was created as a prototypic implementation for this approach. An online application, *BudHat*, furthermore exemplifies the envisioned functionality of a model version control system.

We can only query against that which we have collected. And so if someone has never made a ripple in the pond in Syria in a way that would get their identity or their interest reflected in our database, we can query our database until the cows come home, but there will be nothing show up because we have no record of them.

— *James Comey, former director of the FBI*

6. Accommodation for inhabitants

Combining computational models, semantic annotations, and simulation experiments in a graph database [Henkel et al., 2015] was the first approach towards an integrative storage concept of models and accompanying data. It also changed the idea of model search towards an integrated search that spans the domains of models, simulation and annotation.[1]

Research question: Model repositories such as the BioModels Database, the CellML Model Repository, or JWS Online are frequently accessed to retrieve computational models of biological systems. However, their storage concepts support only restricted types of queries and not all data inside the repositories can be retrieved.

Results: In this Chapter a storage concept that meets this challenge is presented. It grounds on a graph database, reflects the models' structure, incorporates semantic annotations and simulation 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. 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. Furthermore,

[1] see Appendix A.2

this work for the first time enables CellML- and SBML-encoded models to be effectively maintained in one database and showcases how these models can be linked via annotations, and queried.

6.1. Introduction

Model repositories such as the BioModels Database [Chelliah et al., 2013, 2015, Glont et al., 2017, Le Novère et al., 2006, Li et al., 2010], the CellML model repository [Lloyd et al., 2008, Sarwar et al., 2019, Yu et al., 2011, 2015], or JWS Online [Olivier and Snoep, 2004, Peters et al., 2017, Van Gend et al., 2007] offer to the community valuable, curated, and reusable 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. Errors in the models' encoding are more likely to be detected, they can be resolved and documented. Finally, models are distributed in standard formats, e. g., the Systems Biology Markup Language (SBML) [Finney et al., 2003, Hucka et al., 2003, 2019] or CellML [Cuellar et al., 2003, Schreiber et al., 2018], making them immediately available to 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. Semantic annotations relate model entities to external resources describing the underlying biology. For example, a model of the cell cycle may be annotated with a term from Gene Ontology (GO) [Ashburner et al., 2000, Consortium, 2018] defining 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)

Please note that the SBML file would only be equipped with the GO identifier (here: GO:0007049). This identifier can then be resolved computationally to access the full information from Gene Ontology, making a semantic-based comparison of

models feasible, e. g., [Dumontier et al., 2013, Henkel et al., 2010, Schulz et al., 2011, Thavappiragasam et al., 2014].

Over the past years, the focus shifted from encoding pure model code towards providing full model-related information [Le Novère et al., 2005, Scharm et al., 2014, Waltemath et al., 2011a]. As a consequence, researchers have access to detailed descriptions of 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]. A MIRIAM-compliant model contains information about each biological entity, links to the publication describing a model (denoted as reference publication), and instructions on how to use a model to reproduce a published result. Driven by the definition of the MIRIAM requirements and related efforts under the Computational Network for Modeling in Biology (COMBINE [Le Novère et al., 2011]), model repositories provide richly annotated models.

Before the era of semantic knowledge integration and ontologies, model code contained only few meta-data. Models could easily be kept in file systems and meta-data in relational data tables [Li et al., 2010]. A main feature of relational data tables is their fixed database schema which enables fast storage of and search for homogeneous data items. This storage concept has become unsuitable, because today's models contain heterogeneous meta-data. Even MIRIAM-compliant annotations cannot be mapped efficiently onto relational tables. The heterogeneous structure and content of the various meta-data encoded in a model do not comply with the homogenous and pre-defined properties of relational tables. This (technical) limitation led to a situation where many types of model-related data are not extracted from the model. Consequently, the information contained in these meta-data items is not accessible for any kind of comparison or reasoning. Meta-data that is in principle available, but not retrievable include the structure of the model [Henkel et al., 2012a], model versions [Waltemath et al., 2013a], and simulation setups [Waltemath et al., 2013c]. One consequence of inaccessible meta-data is that modeling results may become irreproducible, because the information on the simulation experiment is not associated with the model code [Waltemath et al., 2011b].

This Chapter proposes and describes a concept of graph databases for model storage and retrieval. Graph databases support heterogeneous data structures. They furthermore enable a flexible integration of model-related meta-data. The focus here is on models encoded in SBML and CellML formats, associated simulation setups in SED-ML format, and curated with semantic annotations from bio-ontologies. A key

feature of the concept described here is the *explicit linking of data*. It enables, for the first time, queries across different data formats, e g. "Return experiments observing entities representing an 'm-phase inducer phosphatase' and acting as modifier in a reaction". This query incorporates and links several types of meta-data and creates a complete picture of the model: model code (identifying all models that contain entity X); semantic annotations (identifying all entities X that are annotated with concepts from a bio-ontology that relate to "m-phase inducer phosphatase"); the model's network structure (filtering those models where X is a modifier in a reaction); and simulation experiments (identifying possible simulation setups for the chosen models). This concept, when implemented in open and private model repositories, supports modelers and biologists in retrieving models and scientific findings, fosters the exploration of published models and increases model reuse.

6.2. A graph database for simulation models and associated data

Model reuse can be improved if models *and* meta-data are considered together. In this Chapter presents a novel storage concept that tightly links model code with model-related data. It is directly relevant for developers of model repositories in computational biology, as it offers new possibilities for model search and comparison. This Chapter also elucidates the possibilities of state-of-the-art database techniques for handling the increasing amount of data related to a modeling work.

This work was driven by one question: If models encode networks - why do we not store them as graphs rather than as relational tables? In a graph-database, nodes contain the data and edges represent the links between the data. Graph databases are a suitable technology for the integration and storage of modeling results, 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, model entities, and meta-data are difficult to represent in a table-based relational database management system. 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 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.

Traditionally, relational databases were developed for homogeneous, structured data, e. g., numerical data. 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 business cards that identify the model author and curators [2], or *Dublin Core*, a vocabulary mainly used to describe web resources[3]. Some models are provided with simulation setups and graphical representations. Designing a relational representation for these links and keeping the database efficient at the same time is impossible.

A core concept of relational databases is their fixed schema which defines the structure of the data. Semi-structured documents, however, have only loose constraints on the data structure [Buneman, 1997], which cannot be handled efficiently by relational databases [Robinson et al., 2013]. As all standards for model encoding are semi-structured, relational databases are not the best choice for efficient storage.

NoSQL approaches, together with semantic web applications, more recently gained popularity in the life sciences [Splendiani et al., 2012], e. g., as Key-Value Stores, BigTable [Ghemawat et al., 2003], document databases, triple stores, or graph databases [Angles and Gutierrez, 2008]. We chose the graph database Neo4j [Robinson et al., 2013, Vicknair et al., 2010]. It represents 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).

6.2.1. Incorporated data domains

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., 2013c]. 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 [Cuellar et al., 2003, Henkel et al., 2016a, Hucka et al., 2003, Le Novère et al., 2009], simulation descriptions [Waltemath et al., 2011c], simulation results [Dada et al., 2010] and semantic annotations [Demir et al., 2010, Gennari et al., 2011, Waltemath et al.,

[2] vCard: `http://www.w3.org/TR/vcard-rdf/`
[3] Dublin Core: `http://dublincore.org/`

2011e]. Here, the focus is on the data requested by two Minimum Information Guidelines: MIRIAM [Le Novère et al., 2005] for requested information about models and MIASE, the Minimum Information About a Simulation Experiment [Waltemath et al., 2011a], for requested information about simulation setups.

The development of standards is a continuous process, and their uptake by software tools and users progresses at varying speed. For example, while many journals today recommend, or require, the provision of model code during submission (e. g., in SBML), there is rarely a 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 model reproducibility have been published in the past years [Le Novère et al., 2005, Waltemath and Wolkenhauer, 2016, Waltemath et al., 2011a]. The ongoing 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., 2014] or the COMBINE Archive [Bergmann et al., 2014, Scharm et al., 2014]. In the following, types of data that have been formally specified and for which curated data is available are considered. These are basically the model code, simulation descriptions, semantic annotations and cross-references, and the mathematical characterization of models [Waltemath et al., 2013c].

Model code in public repositories Modelers predominantly implement their models in native programming languages, most commonly C or C++; script languages such as MATLAB, R or Python; and using graphical representations. 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. BioModels Database, for

example, guarantees persistence and long-term availability of 548 curated models[4] 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].

Simulation descriptions The ability to represent increasingly complex biological phenomena requires models to be instantiated using different initial conditions and parameters, and these conditions must be formally described together with the model itself [Cooper et al., 2014]. 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. The more recent Level 1 Version 2 covers a broader range of experiments, including pulse experiments and parameter scans [Bergmann et al., 2013]. BioModels Database and the CellML Model Repository provide SED-ML files for selected models in their repositories. Surprisingly though, these SED-ML files are not linked with the models inside the databases. Consequently, the information about applicable simulation experiments for a model is not computationally accessible.

This information 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 simulation setups are a kind of meta-data that is relevant for 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 [Fergerson et al., 2015, Noy et al., 2009]. Both SBML and CellML use bio-ontologies to enrich model descriptions with semantic annotations, using RDF triples [Lassila et al., 1998b]. In BioModels Database,

[4] release 28 of BioModels Database as of September 16th 2014. The data for 2019 is shown in Chapter 8.1

models carry between three and 800 annotations, but on average 71 annotations, per model [Alm et al., 2014, 2015]. For example, an annotation could be added to the SBML species 'X', linking it to the ontology term 'ATP' in ChEBI [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.

6.2.2. Database design and data import

The following section describes the structure behind a prototypical graph-database setup. For demonstration purposes, one of the early Tyson models on cell division [Tyson, 1991] is used. This model is fairly small, it is available in SBML (BioModels [5]) and CellML (CellML Model Repository[6]) format.

A so-called document node is created for each model or simulation description item. This is done in concordance with the document-centric version control approach described in Chapter 5. Attached to this node can be a model code (e. g., an SBML node) or model-related data item (e. g., a SED-ML node). The entry point for each ontology is a so-called ontology node.

More specifically, **SBML models** (Figure 6.1, left) are represented by a model node which stores the model's name (in cyan color) and identifier. Attached to the model node are annotation nodes, including the reference publication (purple and gray). 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 6.1 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`. The qualifiers in the SBML model [Le Novère et al., 2005, Waltemath et al., 2011e] allows to incorporate further information on the type of relation between an entity and an ontology concept, e. g., that `pM` is linked to `Cell` via the relation `isContainedIn`.

[5] Tyson1991 - Cell Cycle 6 var at BioModels: `http://www.ebi.ac.uk/biomodels-main/BIOMD0000000005`

[6] Modeling the Cell Division Cycle:cdc2 and Cyclin Interactions at CellML Model Repository: `http://models.cellml.org/exposure/9bff394be3ade829feed94151b3d68b3/tyson_1991.cellml/view`

Further model entities are stored analogously, i .e. encoded parameters, events and other SBML concepts. For example, for each global parameter a node is created and attached to the model node. The same procedure holds for functions. Only Unit declarations are currently omitted, They may be implemented in future versions of the graph database, e. g., for applications such as model merging. 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 the example used here, the species node `pM` is related via `hasPart` to the InterPro term `Diphthine synthase` (in gray). Taken together, the sum of extracted information provides a detailed representation of the models' network structure and all annotations.

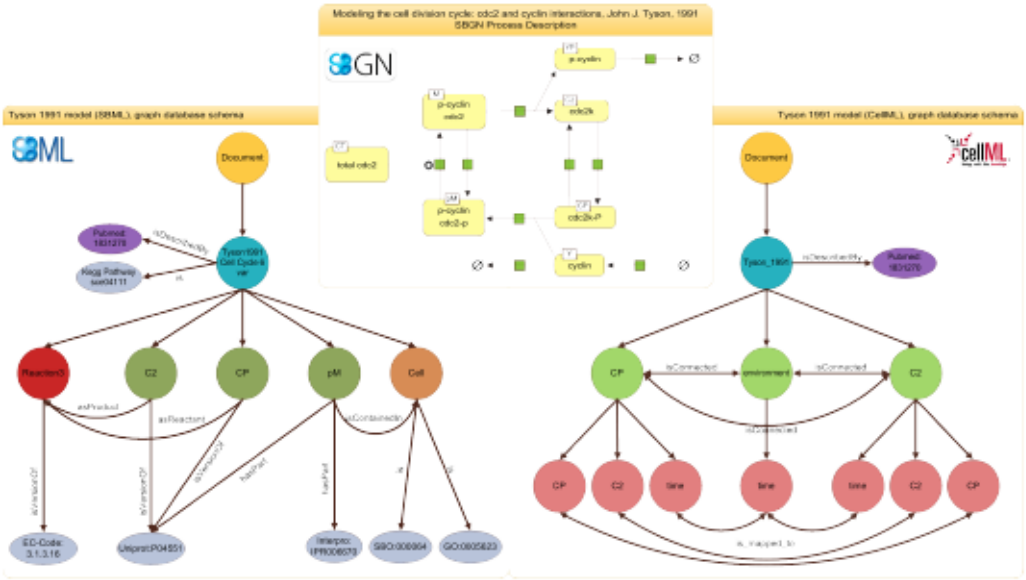


Figure 6.1.: *Representations of the Tyson 1991 model.* The SBGN (top) representation shows the process description for the Tyson 1991 model. The representation of the SBML model inside the graph database is shown on the left, the representation of the CellML model is shown on the right. The document 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. The Figure shows only an excerpt of the model representation, for example many nodes and edges are omitted in favor for readability.

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 that serves as an anchor for the component nodes. Each model entity can be related to a semantic annotation. Figure 6.1 (right) shows the CellML representation of the aforementioned 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 fully annotated CellML models [de Bono et al., 2011, Gennari et al., 2011, Wimalaratne et al., 2009].

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 the design described here, the `SEDML` node serves as the anchor for an experiment. The `Modelreference` node links the experiment to all `Model` nodes used in the simulation. Figure 6.2 exemplifies how a model reference links one SED-ML description to an SBML and a CellML model. Algorithms used for simulation are described by concepts from the Kinetic Simulation Algorithm Ontology (KiSAO) [Courtot et al., 2011], which is one of the bio-ontologies that imported into the graph-database. A subset of KiSAO terms is depicted in Figure 6.2.

Incorporating the concepts of frequently used **bio-ontologies** allows to to query the information hidden in the semantic annotations of in model representations and simulation descriptions. For example, model entities are mostly annotated with concepts from SBO, GO, ChEBI [Alm et al., 2014, 2015]; 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. These ontologies are parsed and all concepts and relations are added as nodes and edges, respectively. Cross references between concepts of different ontologies are currently not mapped to the database as this

Data domain	Documents	Nodes
SBML	462	91488
CellML	841	143521
SED-ML	38	3352

Ontology	Nodes	Domain references
KiSAO	261	38
SBO	606	8839
GO	39787	7555

Table 6.1.: Nodes and documents per Ontology and Domain
Nodes and documents per Ontology and Domain. Top: Number of files and stored nodes for each data domain. Bottom: 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, KiSAO concepts are only linked to the domain of SED-ML while SBO and GO concepts only refer the CellML or SBML domain.

requires a thorough pre-processing and is error-prone [Schulz et al., 2011, 2012] .

Table 6.1 summarizes the data types, number and size of the documents in the database. The integration of further data resources is possible using import tools. An example is, the generic importer for ontologies encoded in OWL which is part of the prototypic implementation. However, the links between the incorporated ontology concepts, models and model-related data need to be defined manually. Adding data encoded in SBRML [Dada et al., 2010] would require additional importers and again a manual post-processing.

6.2.3. Linking model-related data

The main advantage of the previously described concept is its possibility to define flexible links between the data domains. In concordance with previous considerations [Henkel et al., 2012a, Waltemath et al., 2013c], the following types of links are incorporated:

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

(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 6.2 depicts all links for the Tyson model. The database contains the SBML

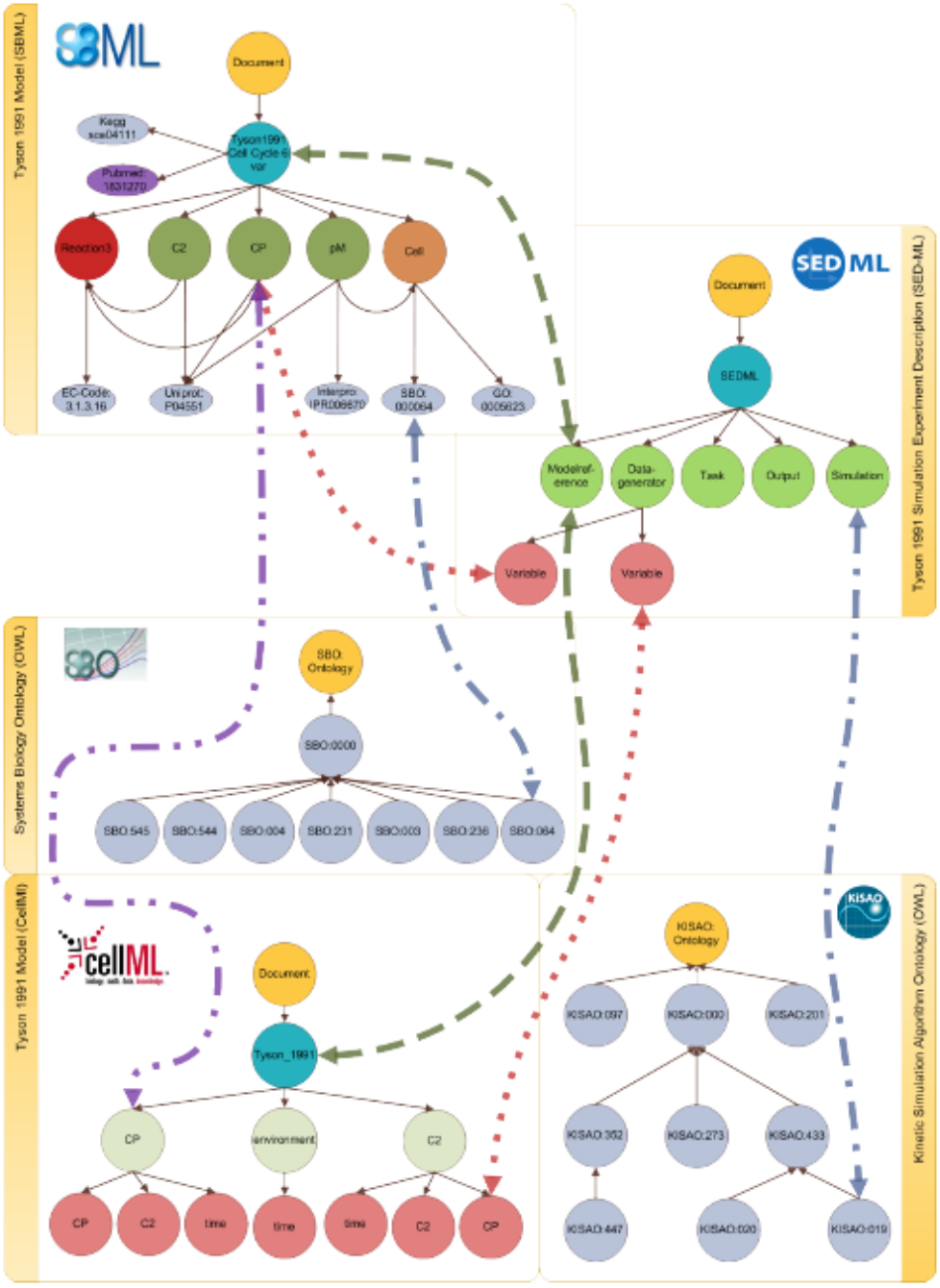


Figure 6.2.: *Linking models, simulation descriptions and ontologies.* The linking between different data domains is shown: simulation experiment descriptions and models (dashed line); defined observation variables and model entities (dotted line); annotated model entities and simulation experiment descriptions (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. All annotations referring to GO are mapped, but not shown in the figure.

and CellML representations of this model. 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, only existing annotations are considered. For each annotation in a model an explicit link to the data entry in the referenced bio-ontology is added. For example, based on the SBO annotation in the SBML model an additional edge between the node representing that annotation in SBML and the entry in SBO itself is created. 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, the model references are resolved. Subsequently it is checked if those models are contained in the database. Vice versa, if a model is added, it is checked if a fitting simulation description exists. 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 6.2, the original SED-ML file contained two model references, pointing to the Tyson 1991 model in SBML and CellML format, respectively. Thus two edges are added.

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 the specific processing of a model entity is not stored, the information if a model entity is part of a simulation is kept. A third type of link thus relates Datagenerator nodes with model entities (c) when a SED-ML file is imported. Also, species that are altered during SED-ML pre-processing are flagged (e. g., if the concentration of a species is changed).

The links (a) – (c) can be inferred from information encoded in the models. They are therefore regarded as explicit links. In addition, implicit links between models of different representation formats are determined (d). As shown earlier, two models may link to the same publication (`Pubmed:1831270` in Figure 6.2). If two models share a publication, the systems can infer implicit links between those entities that are equally named. Entities with similar names (e .g. in terms of Levenshtein Distance or stemming) also have a high probability of being identical. The confidence can be increased further if the entities' annotations match. Figure 6.2 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. Please be aware that linking two models based on the ideas described above does not necessarily mean these models are equal. It only means that they

are similar. If and when models can be considered equal is an ongoing discussion that is not part of this thesis.

6.3. Discussion

6.3.1. 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.

Currently, models and model-related data are only sparsely linked in the, predominantly relational, model repositories. Relational databases store data in tables and use the concept of primary and foreign keys, making them a strong tool for the storage of structured, homogeneous data. On the contrary, they do not perform well on highly connected, semi-structured and heterogeneous XML-based model 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, the prototypic database contains two representations of Tyson's 1991 cell cycle model, in SBML and in CellML, respectively. The two models are connected via an edge between the model nodes. 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, the 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 6.2). Finally, the database establishes links from model annotations into bio-ontologies. For example, the SBML model in Figure 6.2 contains the entity `Cell` which is annotated with a term from SBO. Thus, it is easy to retrieve all models that are annotated with a particular ontology term. This is, for example, helpful in the classification of models as shown in Alm et al. [2014, 2015].

Neo4j as the used graph database implementation is schema optional. Thus new data resources can efficiently be integrated and the database easily be extended.

Subsequently the storage concepts can be extended to store additional information. An example are diffs created by *BiVeS*[Peters, 2016]. 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 6.1 shows how to keep the information on the model structure: For each reaction in the model all reactants, modifier and products are mapped. Participants in the reactions are furthermore linked to the bio-ontologies, to simulation setups, and linked across the SBML and CellML representations. Building on this structured representation, a query can now add restrictions on the reaction network, e. g., formulating 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. For this specific example, the prototypic graph database retrieves the model "Schaber2012 - Hog pathway in yeast" [7], because the species `Hog1PPActive` occurs in ten reactions and only acts as a modifier (Query 1).[8]

```
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 using the ranked retrieval system [Henkel et al., 2010] as described in Chapter 4 and 8.1.1.

6.3.2. Exploiting links to associated virtual experiments

"Which simulation experiments in the repository investigate the change of concentration in 'm- phase inducer phosphatase'?". To answer this question, it is not sufficient to query just the model domain. The retrieval system must also incorporate

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

[8] The following examples are based on BioModels Release 25.

information about, and links to, simulation experiments. Sometimes open repositories supplement models with SED-ML files, enabling users to reproduce one or more figures of the reference publication. However, current model repositories do not explicitly store the link between model and simulation description. Consequently, one cannot retrieve relations of that kind. For example, Novak's 1997 cell cycle model can be run with at least two different setups, either reproducing Figure 2a or Figure 2b of [Novak and Tyson, 1997]. The implementation described here keeps the links between simulation setups and models and thus knows which experiments are applicable to which models. Query 2 is an example for the retrieval of all known simulations for a model.

```
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]

The links between SED-ML elements and KiSAO allows to define restrictions on the SED-ML files to be considered in a search result, e. g., to retrieve only models that can actually be simulated with a given simulation algorithm. Furthermore, the query "Which CellML encoded models can be simulated using a Livermore Solver?" can be answered (Query 3).

```
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.

One can also imagine to restrict results to changes in concentrations of a certain parameter. Finally, SED-ML descriptions may be defined as templates for virtual experiments. Virtual experiments are *in silico* assays of a model's behavioral repertoire, both in declaring what a model should do and verifying what it actually does [Cooper et al., 2014]. Such simulation descriptions are per definition applicable to classes of models, enabling the clustering of models by type of experiment that they reproduce correctly. 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.

```
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 6.3.

It retrieves a 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 6.3. To the best knowledge this approach is the first able to answer queries spanned over different data sets and combining them with an index look-up in the field of computational biological models.

6.3.3. Statistics

Provide interesting statistics about models is also within the capabilities of this prototypic implementation. For example, one can identify the term SBO:0000009 (kinetic constant) as the most frequently used annotation in BioModels Database (Query 5) and compute the number of annotations using SBO:0000009 or one of its 125 children (Query 6).

```
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 (kinetic constant, used 1127 times), SBO:0000252 (polypeptide chain, used 509 times), GO:0043241 (protein complex disassembly, used 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 (kinetic constant) or one of its children?
Result 6: 3373 annotations pointing to SBO:0000009 or one of its children (e. g. half-life, diffusion coefficient or bimolecular rate constant), 1127 of them point directly to SBO:0000009.

6. Model Storage

Finally, the database can derive statistical values. For example, the average 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, respectively (Query 7).

```
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.

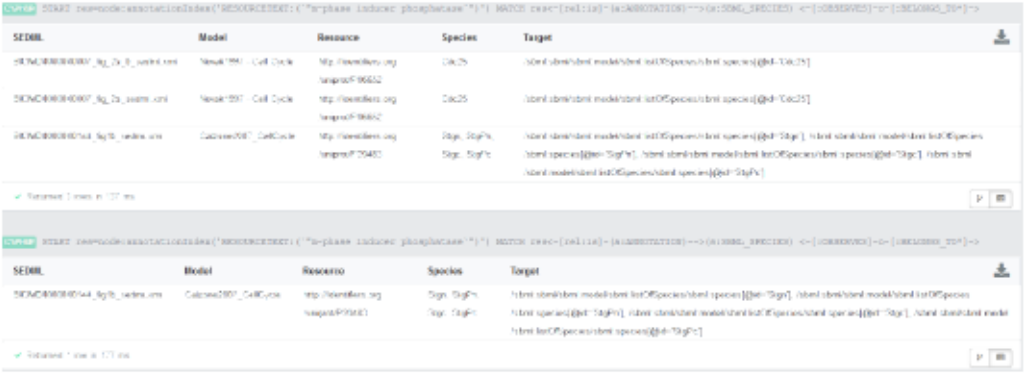


Figure 6.3.: *Results for Query BM3.* The query output at the top of the figure restricts the species role to *modifier*. Three SED-ML files match. The first and second file belong to the same model and both observe the species *Cdc25*. The third query result is a SED-ML file observing four different species. The query output at the bottom of the figure shows the result of a similar query. Here the species must act as *reactants*. Only one SED-ML file is retrieved, namely the third result of the top query. All retrieved species (declared as observed by a SED-ML file) are annotated with a UniProt ID. The annotation is either *P06652*, the protein *Cdc25* in yeast, or *P20483*, 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 corresponding to 'm-phase inducer phosphatase', the database would have also returned the simulation description for that model.

6.3.4. Comparison with other approaches

RDF-triple-stores and SPARQL: Semantic systems biology has been termed the 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 convert model representations into semantically enriched formats to compare models and to improve the integration with knowledge in bio-ontologies [Hoehndorf et al., 2011, Wimalaratne et al., 2014]. In

general, the idea is to transform all data into RDF representations, store the RDF triples into databases, and provide SPARQL[9] endpoints to access the triples. For example, SBML models in BioModels Database can be converted into OWL or RDF representations, using straight forward to more complex transformation methods [Hoehndorf et al., 2011, Köhn and Strömbäck, 2008, Lister et al., 2010, Wimalaratne et al., 2014]. 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[10] [Belleau et al., 2008] or the BioModels Database SPARQL endpoint [Jupp et al., 2014]. While many formats can be transformed to RDF, an RDF representation is not available for all data in the approach described in this Chapter (i.e., SED-ML). An in-depth comparison of graph-databases with RDF triple-stores (and associated query languages) is not in the scope of this thesis' Chapter. The major reason a graph database in this thesis was the support for graph algorithms. Triple-stores and SPARQL are tailored towards sub-graph retrieval. However, common graph algorithms such as 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]. The main argument here is, that the graph structure and thus graph algorithms will become more important in the domain of computational biology. A first example showcasing the use of graph algorithms for computational biology models is the extraction of characteristic features for sets of thematically similar models (e.g., cell cycle or NF-κB) [Alm et al., 2014, 2015].

BioModels Database: Recently, the European Bioinformatics Institute in Hinxton, UK, announced a SPARQL endpoint for BioModels Database. All SBML encoded models in BioModels Database were converted into RDF representations and added to the EBI RDF Platform [Jupp et al., 2014]. For comparison of the concept described in this chapter against BioModels Database's approach, a translation of the query examples from the EBI web page[11] into queries, compatible with the implementation used here, was done. The executed queries and retrieved results are shown in the following three listings. Additionally, Figure 6.4shows a visualization for Query BM3.

[9]SPARQL: `http://www.w3.org/TR/rdf-sparql-query/`
[10]Bio2RDF: `http://bio2rdf.org/`
[11]EBI SPARQL Endpoint: `https://www.ebi.ac.uk/rdf/services/sparql`

```
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 6.4.

Translation and application of all queries and reproduction of the results was easily possible. Specifically, query BM3 asks for models annotated with the 'acetylcholine-gated channel complex'. Due to the missing index support for this RDF store, the user must first manually look up and transform this annotation term into a URL, and paste the URL into the query. On the contrary, the graph-database 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 and in Query M6.

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

The COMBINE archive is well suited for the export and exchange of research results. The graph database rather offers a solution for the management of those files. For example, the CombineArchive Toolkit [Scharm et al., 2014] may query a database crated using the concepts described here, collect all necessary information

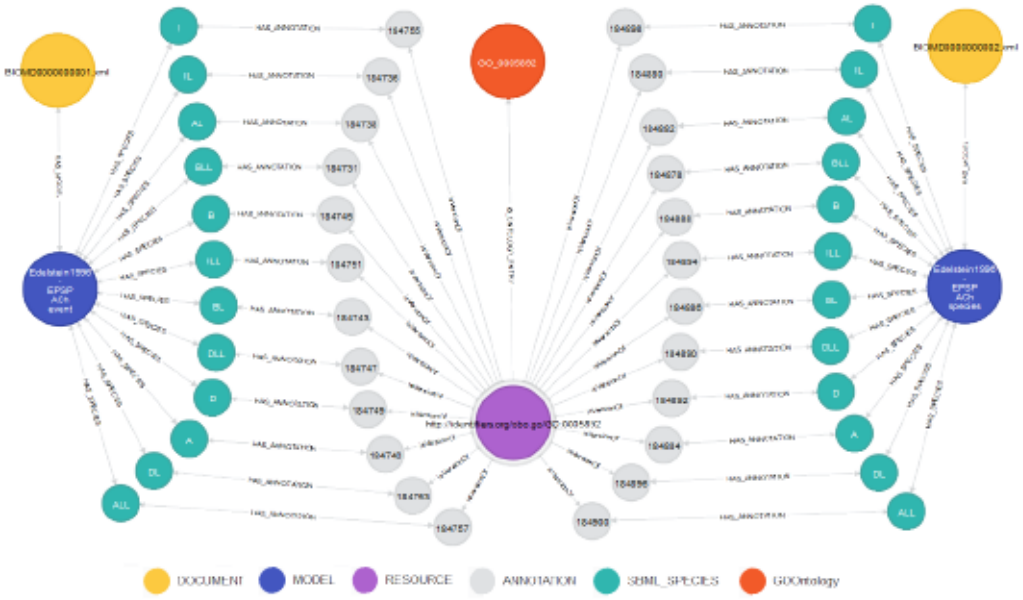


Figure 6.4.: *Visualization of Query BM3.* The centered node (purple) is the requested annotation *GO:0005892* (acetylcholine-gated channel complex). This node is connected to the orange node representing the GO term within the Gene Ontology. The green nodes represent the species linked to the *GO:0005892* annotation, the blue nodes represent the models "Edelstein1996 - EPSP ACh event" (BIOMD0000000001) and "Edelstein1996 - EPSP ACh species" (BIOMD0000000002). Documents are displayed as yellow nodes. Nodes in gray represent annotation containers as stated in the SBML specification [Hucka et al., 2010] and do not carry any meaning themselves.

and automatically create an Open Modeling EXchange format (OMEX) file[12].

6.3.5. Conclusion

Open model repositories are frequently queried for computational models describing particular aspects of biological systems. However, their storage concepts are restricted and not all data contained inside the repositories is incorporated into the search process.

The concept and prototypic implementation described in this Chapter 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. This Chapter demonstrates how relevant meta-data can be stored in a graph-database using the example of Neo4j. Furthermore it exemplifies

[12]Open Modeling EXchange format: `https://co.mbine.org/standards/`

how this meta-data can subsequently improve model retrieval and thus model reuse.

The concept is easy to adapt and implement. An interface to test and query the database described in this paper is available.[13] In addition, a web API[14] 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.[15]

6.4. Materials and Methods

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

The entry point for each model import is a document node. It 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` and inherently describe a models hierarchy. Furthermore, it is possible to attach an annotation to each model entity, describing the particular entity in more detail. All such annotations are stored in the database and are 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

[13] The original testing instance at `https://sems.uni-rostock.de/projects/masymos/` is no longer available as the SEMS project is terminated. For testing queries and interfaces the CovidGraph (see Chapter 8.5) can be used at `https://db.covidgraph.org/browser/`

[14] The original testing instance at `https://sems.uni-rostock.de/projects/morre/` is no longer available as the SEMS project is terminated. The web api is now incorporated in CovidGraph (see Chapter 8.5) `https://db.covidgraph.org/browser/`

[15] Staging server for Physiome Model Repository `http://staging.physiomeproject.org/`, might not be available at all times.

annotation. Labels are indexed and allow to select all nodes of a specific type.

6.4.2. Implementation

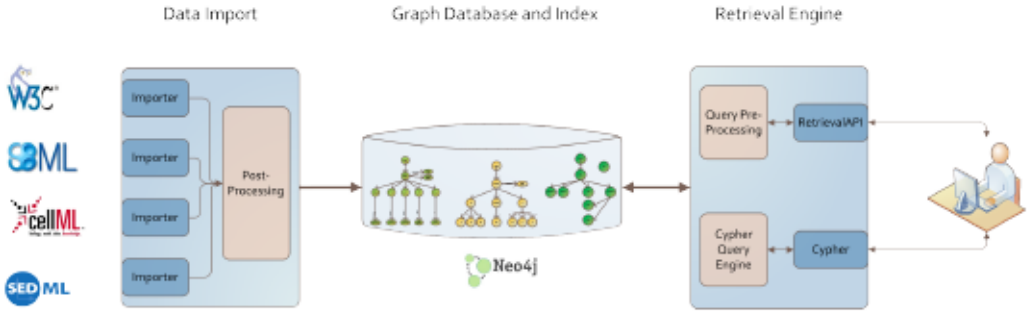


Figure 6.5.: *Architecture and data flow of the graph database.* Data from different models, simulation descriptions or ontologies is imported using format dependent importers. Each import undergoes a post processing afterwards. The stored graph and index structures are available via two retrieval interfaces: Cypher and an adaption of [Henkel et al., 2010]. Both are based on RestAPIs. The data itself is stored in a Neo4j graph database.

The implementation of the graph-based storage corresponds to the architecture depicted in Figure 6.5. The Neo4j[16] 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[17], 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[18] and

[16]Neo4j: `http://www.neo4j.org/`
[17]Apache Lucene: `http://lucene.apache.org/core/`
[18]Owl-api: `http://owlapi.sourceforge.net/`

the JFact[19] 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[20] [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, match and link 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 '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.

6.4.3. Supported types of queries

The Cypher Query Language provides direct access to read and write data. Cypher is the declarative language to pose queries against graph structures, similarly to SQL for relational databases. Using Cypher, the prototypic implementation 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: Examples for database look-ups and filtering are shown in Query M1 and M2. The *MATCH* clause uses a build-in index to retrieve all nodes

[19] JFact: `http://jfact.sourceforge.net/`
[20] Cypher: `http://www.neo4j.org/learn/cypher`

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 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 are given in Table 6.1 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: Cypher also supports statistical queries. Query M5, for example, returns the minimum, maximum and average number of variables attached to components in

CellML models. To provide these statistic 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)

6.4.4. Database scaling

Büchel et al. [2013] describe how to build computational models from biochemical pathway maps. The path2models[21] project resulted in more than 140.000 SBML models of a total size of 70 GB. This data-set was used 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 83 GB, including the indices.

[21] Path2models project: `https://www.ebi.ac.uk/biomodels/path2models`

Part of what it is to be scientifically-literate, it's not simply, 'Do you know what DNA is? Or what the Big Bang is?' That's an aspect of science literacy. The biggest part of it is do you know how to think about information that's presented in front of you.

— *Neil deGrasse Tyson*

7. A model's nature

Notions of similarity for systems biology models [Henkel et al., 2016a] explains different points of view, so called "aspects", of a model. Six aspects potentially relevant for model understanding and comparison are defined: underlying encoding, references to biological entities, quantitative behavior, qualitative behavior, mathematical equations and parameters, and network structure.[1]

Research question: Systems biology models are rapidly increasing in complexity, size and numbers. When building large models, researchers rely on software tools for the retrieval, comparison, combination and merging of models, as well as for version control. These tools need to be able to quantify the differences and similarities between computational models. However, depending on the specific application, the notion of 'similarity' may greatly vary. A general notion of model similarity, applicable to various types of models, is still missing.

Results: This Chapter presents a survey of existing methods for the comparison of models, introduce quantitative measures for model similarity, and discuss potential applications of combined similarity measures. To frame model comparison as a general problem, it describes a theoretical approach to defining and computing similarities based on a combination of different model aspects. The six aspects that are defined as potentially relevant for similarity are underlying encoding, references to biological entities, quantitative behaviour, qualitative behaviour, mathematical

[1] see Appendix A.2

equations and parameters and network structure. Future similarity measures will benefit from combining these model aspects in flexible, problem-specific ways to mimic users' intuition about model similarity, and to support complex model searches in databases.

7.1. Introduction

'Over the past few decades, mathematical models of molecular and gene networks have become an important part of the research toolkit for the biosciences' [Le Novère, 2015]. Mathematical models are formal representations of natural systems that can help answer questions about the complex system they represent [Wolkenhauer, 2014]. According to Robert Rosen, a model establishes a modelling relation between a formal and a natural system: the formal system encodes the natural system, and inferences made in the formal system can be interpreted (decoded) as statements about the natural system [Rosen, 1991]. Systems biology models serve as abstractions of biological systems. Biochemical models, for example, associate model components, such as mathematical expressions, objects or variables, with biochemical entities such as molecule species or chemical reactions. Depending on the scientific questions addressed and on available data, a biological system may be described by models of different scopes and levels of granularity, reflecting different views of the system.

In this article, the focus is on (computational) systems biology models. Systems biology models can be based on a number of mathematical formalisms [Le Novère, 2015]. Here, again, the choice of a particular approach largely depends on the type of question asked and on the data available [Wolkenhauer, 2014]. Metabolic and signalling pathways are usually modelled by ordinary differential equation systems, and the resulting models are known as kinetic models. Larger metabolic systems are typically described by constraint-based network models that capture stationary metabolic fluxes, but disregard enzyme kinetics. Gene expression dynamics can be modelled by kinetic models, stochastic processes or discrete dynamic processes such as Boolean networks [Wang et al., 2012]. Spatial cell models may even involve partial differential equations. In addition, the rise of synthetic biology and the simulation of entire organisms require hybrid modelling.

Because models are rapidly increasing in numbers and in complexity, many of them are now formally encoded in standard formats to be processed by and exchanged among different software tools. The Systems Biology Markup Language (SBML) [Hucka et al., 2003] and CellML [Cuellar et al., 2003] are two XML-based de facto

standards that encode the entities and interactions in biological models. Both enable software interoperability across the diverse landscape of modelling, visualization and simulation software [Hucka et al., 2011]. To further standardize the representation of models and enable interoperability between them, the biological meaning of model components can be specified by semantic annotations that relate components (e.g. a variable representing glucose concentration) to common identifiers defined in ontologies or public databases (e.g. CHEBI:17234, which is the identifier for glucose in the ChEBI database [Degtyarenko et al., 2008]). Together, standard formats and semantic annotations foster the reuse of data in the biosciences [Sansone et al., 2012]. In addition, the reuse of models across research groups and scientific use cases demand sophisticated model management strategies for storage, search, retrieval, version control and provenance [Waltemath, 2015].

All these tasks require, at least implicitly, mathematical notions of similarity between models. If similar models can be recognized by software, then they can also be found in databases, differences between models can be assessed and genealogies of model versions can be reconstructed. For any domain of knowledge, different notions of similarity can be used depending on the aspects between which similarity is determined. For example, to decide whether two persons are alike, one need to choose what features of the person to focus on, and one may assign priorities to selected features. Then, two persons may be compared by how tall they are, by the shapes of their faces, or by their behaviour — with different results. Certain groups of people may be easily distinguished by size (e.g. children versus adults), while the same criterion will fail to predict other distinctions (left-handed versus right-handed people). Also in computer science, data objects can be compared with regard to different aspects depending on the purpose of the comparison. For example, when comparing image files, typical features are the colours used, the objects shown or file size and type. The choice of the features and similarity measure depends on the intended application.

Model comparison has been implemented in a few software tools as part of their similarity searches. For example, public model repositories such as the CellML model repository [Yu et al., 2015] can use a retrieval system [Henkel et al., 2010, 2015] that supports users in finding models relevant to their research. For such a model retrieval system, similarity measures are applied and adapted to yield a measure of model similarity based on various features of models [Lange et al., 2014]. Another example is semanticSBML [Krause et al., 2010], an online software tool that provides functionality for clustering, merging and comparison of SBML models based on

semantic annotations.

The argument here is, that model comparison, as a computational task, should be abstracted from specific model types, applications and software. This Chapter sets out to study what is common to different similarity measures for systems biology models, and attempts to place them into a common mathematical frame. The practical, long term, aim of such an approach is to build a common software library that implements various similarity measures and can be integrated into systems biology software to provide flexibility in model comparison tasks. This Chapter also provides an overview and categorizes existing similarity measures between models and their components. These measures may rely on aspects such as the biological entities described, network structure, model assumptions, mathematical statements and parameter values, or the dynamic behaviour displayed in simulations. The different aspects are discussed and it is shown how they can be incorporated into computable similarity measures. Furthermore, it shows applications for these measures using model search, clustering and merging as examples, and it discusses open problems that should be addressed in future research.

7.2. Formal notions of similarity

7.2.1. Similarity measures

The general notion of similarity can be conceptualized by mathematical functions called similarity measures. Intuitively, a similarity measure, for some type of objects, assigns to each pair of objects a similarity value. Larger values signify greater similarity. If properties of the objects are represented in some property space, similarity resembles an inverse distance: objects with small distances are considered similar, objects with large distances dissimilar. Similarity measures are often normalized to yield values between 0 and 1. A similarity $\sigma = 1$ then implies that two objects are identical (or indistinguishable) with regard to the properties considered, while entirely different objects will have a similarity $\sigma = 0$. Formally, a normalized similarity measure for a set X is defined as a function which assigns $x_1, x_2 \in X$ a value $\sigma \in [0, 1]$. This σ is called the 'similarity value'. Most similarity measures are symmetrical and satisfy the triangle inequality.

7.2.2. Model similarity based on single aspects

Similarity measures for models refer to specific model aspects. For example, the similarity between two network models can be determined by aligning the networks and assessing their common overlap [Clark and Kalita, 2014]. The result is a similarity value with respect to network structure. Alternatively, the similarity between two models can be calculated by comparing simulated time series [Tapinos and Mendes, 2013], by comparing semantic annotations [Henkel et al., 2010] or by identifying occurring patterns [Lambusch et al., 2018], etc. In the following, similarity is defined with respect to certain model properties. However, in the context of this Chapter, similarity is defined with regard to a single aspect. Subsequently, similarity with regard to a combination of aspects is discussed.

To formally express similarity with regard to a single aspect, it is necessary to introduce the projection of a model M onto an aspect α, for example, the projection of a dynamical model onto the underlying network topology. All other model features, which do not belong to this aspect, are ignored. The similarity measure between the two projected models then determines the similarity of these models with respect to that aspect (see Figure 7.1). More formally, for a model aspect α the α-similarity between two models M_1, M_2, $sim_\alpha(M_1, M_2)$ is defined as $\sigma_\alpha(\Pi_\alpha(M_1), \Pi_\alpha(M_2))$, where Π_α is the projection of a model onto aspect α and σ_α is a similarity measure for aspect α. This scheme applies to both, models as mathematical objects and to encoded models, i.e. text files representing these models, and allows for statements such as 'model 1 and model 2 are similar with respect to aspect α'. This definition may leave some similarity measures undefined because the projection function Π_α is considered as a partial function. If a model lacks the aspect for which the similarity measure is defined, the similarity to other models remains undefined as well.

With the same scheme, models can also be compared with other types of data, e.g. to sets of experimentally measured substance concentrations. To see whether a model and a data set refer to similar sets of substances, one may choose the sets of substances as the aspect the current focus is on. Mathematically, if a projection $\Pi_\alpha(M)$ yields the aspect α of a model M and a related projection $\bar{\Pi}_\alpha$ projects data sets D to the same aspect α, then an α-similarity function σ_α can be used to find data sets that resemble a model M with respect to α, by computing $\sigma_\alpha(\Pi_\alpha(M), \bar{\Pi}_\alpha(D))$.

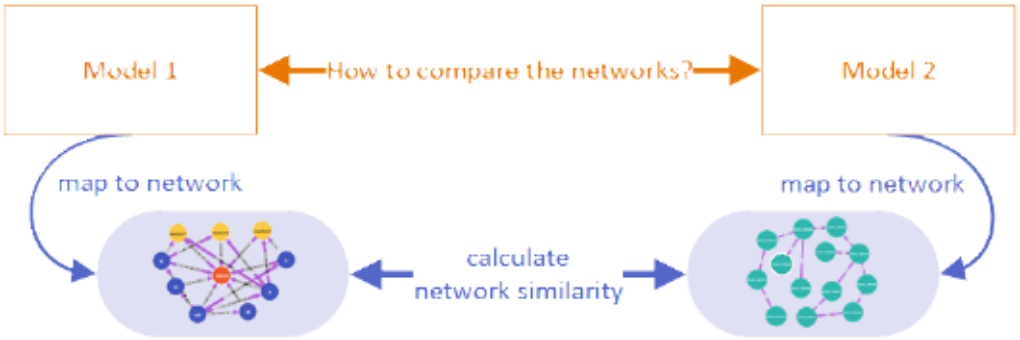


Figure 7.1.: *Comparison of two systems biology models based on a model aspect.* To compare two models, the models are mapped to a particular aspect (in this case, their network structure). Then, an appropriate similarity measure (e.g. for graph similarity) is applied [Clark and Kalita, 2014].

7.2.3. Combined similarity measures

Similarities arising from different model aspects can be combined to define more complex measures. For example, two models may only be considered similar if they describe similar biological entities and if they describe them by similar mathematical formulae. The combination of different similarity measures occurs frequently in model search, e.g. to enable users to find 'models that describe aspects of the cell cycle and use similar parameter space'.

To perform such complex tasks, it is necessary to aggregate multiple similarity scores into a single score. One can compose or decompose complex similarity measures by projecting models on individual aspects and then combining the resulting similarities between these aspects. Formally, this can be expressed as follows (without loss of generality limited to normalized similarity measures): $sim_{\alpha\circ\beta}(M_1, M_2) = \sigma_{\alpha\circ\beta}(\Pi_{\alpha\circ\beta}(M_1), \Pi_{\alpha\circ\beta}(M_2)) = \sigma_\alpha(\Pi_\alpha(M_1), \Pi_\alpha(M_2)) \circ \sigma_\beta(\Pi_\beta(M_1), \Pi_\beta(M_2))$ where $\circ$ is a function $\circ : [0,1] \times [0,1] \mapsto [0,1]$.

For larger numbers of properties to be combined, similarity measures can be conveniently defined by feature vectors. A feature vector represents different model properties by a list of numbers arranged in a vector. For example, the elements of the vector may be zeros or ones, denoting the occurrence of certain annotations in the model, or integer numbers denoting the frequencies of certain network motifs in the network. Once models have been translated into feature vectors, a variety of methods from multivariate data analysis can be applied, including supervised and unsupervised classification [Duda et al., 2000]. Simple similarity measures for feature vectors can be defined based on Euclidean distances, the Jaccard index or on normalized scalar products (i.e. the cosine of the angle between feature vectors). To

weigh different features and to account for their known relationships, special metrics can be used, e.g. quadratic forms instead of simple scalar products. This can be useful to account for known logical connections between different features.

7.3. Model comparison based on specific aspects

Relevant aspects of models can be extracted (or calculated) from the encoded models. Depending on the goal of a comparison and on a model's representation, the computable similarity measure is chosen. When models are presented as network graphs, it will be natural to compare them by network structure using a graph similarity measure. For example, the Systems Biology Graphical Notation [Le Novère et al., 2009, Van Iersel et al., 2012] for graphical display supports comparisons of network structural and visual similarity. When only model equations are given, network similarity would be more difficult to recognize even if all necessary information is implicitly given. For example, SBML and CellML for model structure and formulae support comparisons of structural and mathematical similarity. The Systems Biology Ontology (SBO) [Bernasconi and Masseroli, 2019, Courtot et al., 2011] and other ontologies enable semantic similarity comparisons. According to the concept described in this Chapter, the projection function Π_α can be more or less complex to define (and evaluate) depending on the model representation. Consequently, certain similarity measures will be more natural given a certain model representation.

For this thesis similarity measures are classified based on five types of model aspects: (i) model encoding; (ii) biological meaning; (iii) network structure; (iv) mathematical statements and numerical values; and (v) quantitative and qualitative behaviour (compare Figure 7.2 and Table 7.1). Additional 'meta-properties' and provenance information further improve the comparison (e.g. information about file format or year of development).

7.3.1. Model encoding

Models can be directly compared based on their encoding, i.e. a specific file format. If models are encoded as computer programs, e.g. in MATLAB, a comparison on the syntactic level can be performed using tools for difference detection in software code (e.g. diff, see also Chapter 5). As computer programs, in general, do not share a predefined structure, such a comparison does rarely yield satisfactory results.

However, standardized formats such as SBML and CellML have well-defined

Model aspect	Existing measures	Tool support (selection)	Applications
Encoding	Similarity on the XML-encoding (SBML or CellML), Levenshtein distance	BiVeS [Scharm et al., 2015] Unix Diff	Model version control
Biological meaning	Similarity regarding biological meaning of model components	SemanticSBML, Semantic Measures Library and Toolkit Model set comparison, MORRE [Henkel et al., 2015]	Search, retrieval, comparison of model intentions and assumptions
Network structure	Graph similarity, stoichiometric similarity	Cytoscape [Shannon et al., 2003], graph libraries	Model merging, extraction and comparisaon of submodels
Mathematical statements, quantitative and qualitative behaviour	Difference in statements and numbers; correlation between states, time-serieSemantic Measures Library and Toolkit (for SBO annotations)	Compare dynamic behaviour; classification of models (e.g. oscillatory vsglycolysis models)	

Table 7.1.: *Model aspects and related similarity measures.* The table lists examples of similarity measures for the proposed aspects of a model, examples of software tools supporting these measures and practical application cases.

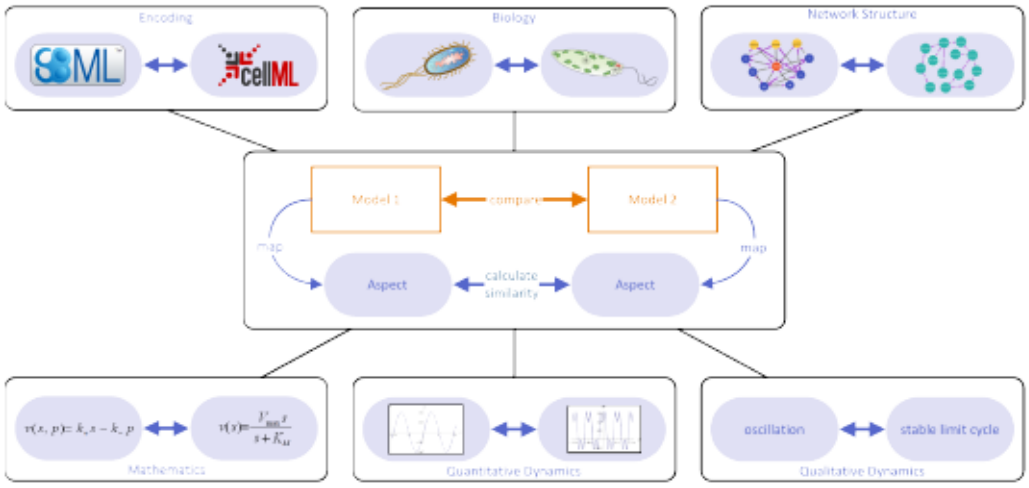


Figure 7.2.: *Similarity measures for models can be derived from similarity measures for model aspects.* Automatic model comparison relies on quantitative similarity measures. A practical way of defining such measures is to project both models to some relevant aspect, e.g. network structure, for which a similarity measure has been defined. In an abstract scheme for model comparison, two models are mapped onto a specific focal aspect.

XML-syntax that restrict the number of possible operations. On the one hand, this constitutes a limitation in what information can be encoded. On the other hand, the limited number of supported structures facilitates the implementation of domain-specific algorithms. These algorithms use the rich information about the encoded biological mechanisms, the components of models and their interactions and the biological meaning of (parts of) the model. Similarly, simulations of models, and the context of the simulation, can be encoded in standard formats such as the Simulation Experiment Description Markup Language (SED-ML [Waltemath et al., 2011b]). Thus, the comparison of models can convey information about their biological meaning and about their behaviour.

Several algorithms use this reduced approach of determining the similarity of models (or model versions) by comparing their syntactic structure, e.g. [Miller et al., 2011, Saffrey and Orton, 2009, Scharm et al., 2015, Waltemath et al., 2013a]. Some software tools compare representations of a model, taking the specific syntactic structure and the corresponding biological meaning into account, while others compare the representations directly and subsequently interpret the results with respect to the biology. For example, XML patches [Saffrey and Orton, 2009] are generated to compare models solely at the XML level, while the BiVeS tool described in [Scharm et al., 2015] (see also Chapter 5 and 8.2) also considers the structure of the actual model representation format.

7.3.2. Biological meaning

To compare models with respect to the biological system described, the meaning of components must be considered. The biological entities in a model (substances, reactions, cell compartments, etc.) must be explicitly associated with a biological description. To this end, biological entities are usually semantically annotated with terms from biological and biomedical ontologies [Courtot et al., 2011]. For example, BioModels provides 1516 literature-based, SBML-encoded models. Among them are 613 curated (manually validated and annotated) models. In addition, BioModels offers 903 non-curated models. A curated model has on average 30 species, 41 reactions, 40 parameters, 16 compartments, 10 rules and 3 events. Each entity in a curated model is on average linked to 1.86 ontology terms providing semantics and biological background[2]. Model annotations may also point to entries in biological databases from which ontology based annotations can then be derived, e.g. UniProt [Apweiler et al., 2004].

Most models in open repositories are annotated with terms from ontologies such as the Gene Ontology (GO) [Ashburner et al., 2000], ChEBI [Degtyarenko et al., 2008] or the SBO [Courtot et al., 2011]. Different ontologies cover different domains of knowledge and allow for model comparison related to these domains: model annotations referring to GO terms allow for defining similarity related to the biological function or cellular location of biological entities; annotations based on ChEBI allow for similarity based on chemical compounds; annotations based on the SBO allow for similarity based on biochemical rate laws or on the roles of chemical compounds in reaction mechanisms, etc. Similarity measures that compare models by their semantic annotations can be defined on the basis of existing similarity measures for ontology elements [Pesquita et al., 2009]. These measures help to identify similarities in model matching and retrieval tasks [Henkel et al., 2010, Pabinger et al., 2014, Schulz et al., 2011, Wimalaratne et al., 2014]. For example, in regulatory pathways describing protein interactions, semantic similarity is traditionally assessed as a function of the shared annotation of proteins with GO terms [Guo et al., 2006].

7.3.3. Network structure

Biological network structures, and especially their statistical properties, have received large attention in systems biology research [Alon, 2006]. Network structures can be extracted from encoded models, and network alignments between models allow

[2] see `http://www.ebi.ac.uk/biomodels-main/news` as of 1 July 2016

for detecting specific structural differences and similarities. Gay et al., for example, proposed graph matching techniques to assess similarities between models [Gay et al., 2010]. They reduced the models to directed, bipartite reaction graphs and searched for epimorphisms between the graph structures. Subsequently, they evaluated their method on SBML models from BioModels. Graphs have also been used to measure the similarity of models based on the motifs they contain [Milo et al., 2002]. Motifs are small partial subgraphs that are statistically overrepresented in a network. Motifs are often interpreted as small functional units, and Alon [Alon, 2003] showed that networks realizing similar tasks (e.g. signalling pathways) contain similar motifs. Therefore, comparing the motif distributions in biological networks could provide information about typical biological functions of these networks [Lambusch et al., 2018].

7.3.4. Mathematical statements and model behaviour

Models can be compared by their quantitative or qualitative dynamic behaviour. This behaviour can either be determined from their mathematical structure or from simulation runs. Similarity of behaviour has been associated with similar internal mechanisms and with dependencies on common extrinsic factors [Tapinos and Mendes, 2013]. While, to the best knowledge, no system is known that formally compares models with respect to dynamic behaviour, there are attempts to compare, for a given model, the simulation results obtained from different simulators [Bergmann and Sauro, 2008]. The Functional Curation framework, for example, compares the dynamic effects of particular simulated perturbations [Cooper et al., 2011, 2016]. A comparison of qualitative model behaviour (e.g. oscillation, steady state) can be useful to identify similarities between biological mechanisms [Tapinos and Mendes, 2013].

Annotations of reactions with terms from SBO may help to determine behavioural similarities in the future. SBO is, for example, already used to annotate mathematic rate laws and therefore a good candidate for a qualitative comparison of mathematical expressions. If mathematical expressions are similar, then the models may also show similar dynamic behaviour.

7.4. Comparison of two models

To illustrate the comparison of models by different aspects, here two dynamical models that stem from the same publication [Tyson, 1991] and study a cyclin and a cyclin-dependent kinase (see Figure 7.3) are analyzed. While the first model describes the phosphorylation of the two compounds by explicit reactions, the second model has been simplified and captures only the general dynamics. The model files, obtained from BioModels (models BIOMD0000000005 and BIOMD0000000006), are encoded in SBML and refer to the organism and biological pathway described. Owing to their different levels of resolution, the models differ in their network structures and mathematical statements. As BIOMD0000000006 is a simplification of BIOMD0000000005 in terms of mathematics, the starting point for this example will be the mathematics aspect.

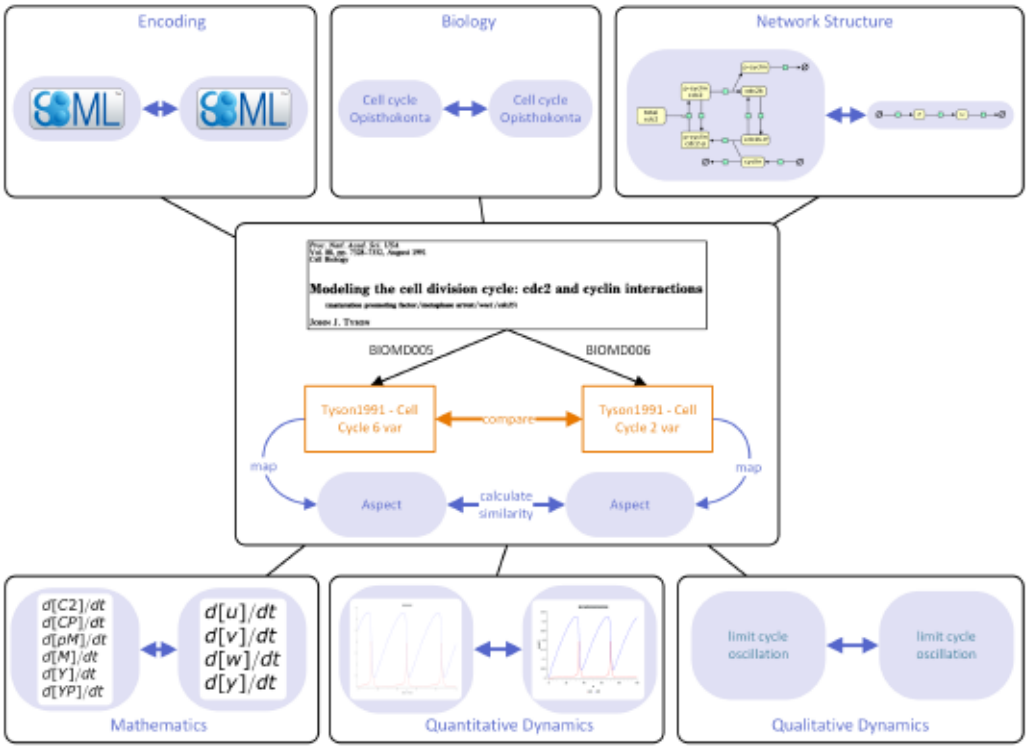


Figure 7.3.: *Comparison of two models describing the cell division cycle.* Two SBML encoded models, obtained from BioModels, are compared by different aspects. Both models stem from the same publication describing the cell division cycle [Tyson, 1991]. Each box represents an aspect and shows an excerpt of BIOMD0000000005 on the left and BIOMD0000000006 on the right side.

7.4.1. Mathmatics

Mathematically, each of the two models is represented by a system of differential equations. The first model encodes to six molecular species (including phosphorylated and unphosphorylated forms of the same compounds) and describes their dynamics by rate equations in which elementary reaction steps are clearly visible. The reaction rates depend on molecular species concentrations, two constant concentrations, and nine rate constants. In the second model, the equations have been radically simplified: by summing and normalizing the variables, one obtains four new variables whose differential equations do not have the form of simple rate equations anymore. In a next simplification step, by an approximation based on time scale separation, the number of variables is further reduced to two, and one is left with four parameters only. Could the similarities between the three equation systems be recognized automatically by inspecting the mathematical formulae? Between the six-variable and the four-variable model, the main problem is the change of variables: even though the second equation system is directly derived from the first one, the correspondence is hard to see if the definition of the new variables (in terms of the old ones) is not known. Especially, the new variable names do not give any clue about their relation to the old variables. Between the four-variable and the two-variable model, by contrast, two variables are abandoned, but the other two variables remain unchanged (same names and same meaning). The remaining two equations are formally similar to equations from the previous model variant, and only some details have changed (e.g. a term being omitted). This formal similarity could be recognized by comparing the formulae in a string or tree representation.

7.4.2. Quantitative dynamics

To compare the simulation results of the two models, one has to introduce two variables in BIOMD0000000005, $[M]/[CT]$ and $([Y] + [pM] + [M])/[CT]$ which correspond to u and v in BIOMD0000000006, respectively. The two time courses are similar except for a phase shift of about $15s$ and a different beginning part (compare Figure 7.4). This shift is owing to different initial conditions. If one uses equal initial conditions the time courses will become identical.

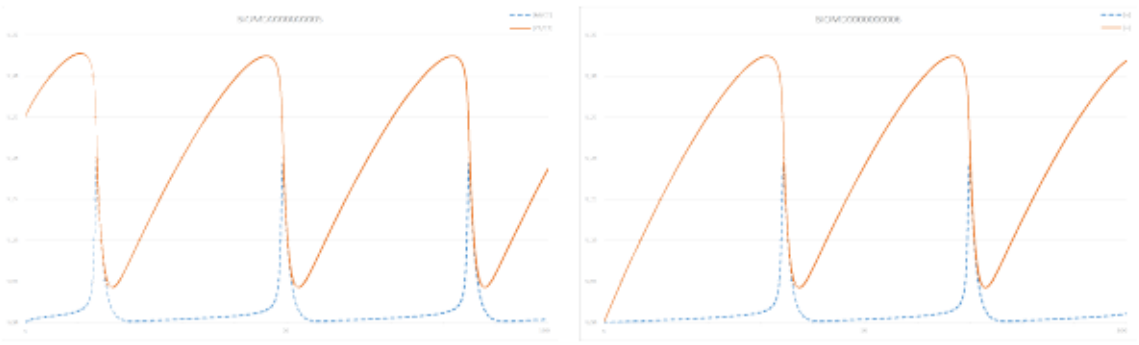


Figure 7.4.: *Time courses for BIOMD0000000005* ***(left)*** *and BIOMD0000000006* ***(right)***. Both time courses (with a duration of 100*s*) were generated with COPASI [Hoops et al., 2006]. Model parameters and initial conditions remain as encoded in the SBML-files obtained from BioModels.

7.4.3. Qualitative dynamics

Both time courses can be characterized as stable limit cycles, i.e. as undamped oscillating behaviour. TEDDY, the TErminology for the Descriptions of DYnamics [Courtot et al., 2011], provides the term TEDDY_0000114 to annotate these qualitative dynamics. See [Knüpfer et al., 2013] for a more elaborate qualitative characterization of the dynamics of both models. A comparison of the qualitative dynamics of both models would reveal that the models are equal in this respect.

7.4.4. Encoding

As described in the aspect Mathematics, the second model contains fewer variables and equations. This is also reflected in the encoding. Both models are encoded in SBML Level 2, Version 4. The BiVeS algorithm for difference detection in computational models can be used to identify changes between the two SBML encodings. The BiVeS output is shown in Table 7.2. It provides information about insertion, deletion, updates and moves to convert one model into the other. A closer look at the table reveals that one model can be considered a simplification of the other. The number and types of identified changes, e.g. characterized with terms from COmputational MOdels DIffer (COMODI), an ontology describing possible changes on computational biology models [Scharm et al., 2016] (see Chapter 8.2.2), could be used as a similarity measure.

Species	Changes
CT (total_cdc2)	deleted
EmptySet	Attribute **hasOnlySubstanceUnits** was inserted: true
	Attribute **initialAmount** has changed: 0 → 1
	Attribute **sboTerm** was inserted: SBO:0000291
YP (p-cyclin)	deleted
Y (cyclin) → v	Attribute **boundaryCondition** was inserted: true
	Attribute **hasOnlySubstanceUnits** was inserted: true
	Attribute **sboTerm** was inserted: SBO:0000297
	Attribute **name** was deleted: cyclin
	Attribute **id** has changed: Y → v
YT (total_cyclin)	deleted
CP (cdc2k-P)	deleted
M (p-cyclin cdc2) → z	Attribute **boundaryCondition** was inserted: true
	Attribute **hasOnlySubstanceUnits** was inserted: true
	Attribute **sboTerm** was inserted: SBO:0000297
	Attribute name was deleted: p-cyclin_cdc2
	Attribute **id** has changed: M → z
pM (p-cyclin cdc2-p)	deleted
C2 (cdc2k) → u	Attribute **boundaryCondition** was inserted: true
	Attribute **hasOnlySubstanceUnits** was inserted: true
	Attribute **sboTerm** was inserted: SBO:0000297
	Attribute **name** was deleted: cdc2k
	Attribute **id** has changed: C2 → u

Table 7.2.: *Comparison of encoded species.* This table lists the changes for the SBML-specific species elements. BIOMD0000000005 encodes nine species, whereas BIOMD0000000006 only encodes six species. For each species in BIOMD0000000005 deletions, insertions or modification are listed.

7.4.5. Biology

Based on two encoded models only, how could one know that the models describe the same biological pathway? Both models carry semantic annotations describing the cyclins and kinases involved. Using the original model BIOMD0000000005 as a query, one can screen BioModels for similar models by using the semanticSBML tool and obtain a ranked list of result models, where the ranks depend on similarity scores computed by assessing the percentage of shared biological annotations. If one runs this query on the manually curated models of BioModels Database, the query model itself appears on top of the list, whereas model BIOMD0000000006 appears only at rank 28. The models ranking above describe cell cycle and MAP-kinase pathways, containing many similar annotations. As BIOMD0000000006 is a simplification with few annotations on a more abstract level, there is only a moderate similarity in terms of biological annotations. In summary, the biological similarity of these two example models can be detected, but there are other models that contain the same annotations and obtain similar, or even higher, similarity scores (in particular, Goldbeter's minimal models of the mitotic oscillator [Goldbeter, 1991]).

7.4.6. Network

A simple dissimilarity measure for networks is the graph edit distance [Gao et al., 2010]. The basic idea is to align two graphs with respect to the lowest edit costs, meaning to insert and delete as few nodes and edges as possible to transform one graph into the other:

$\mathrm{GED}(g_1, g_2) = min_{(e_1,\ldots,e_k)\in P(g_1,g_2)} \sum_{i=1}^{k} c(e_i)$ where $P(g_1, g_2)$ is the set of edit paths transforming g_1 into g_2 and $c(e) > 0$ is the cost of each graph edit operation e. In this example label changes are disregarded. Figure 7.5 shows the necessary edits to transform the network of BIOMD0000000005 into the network of BIOMD0000000006.

7.5. Practical applications of similarity measures

To profit from the wealth of published models, modellers need software that helps them explore, access, compare, simulate and combine models with minimal effort. Here, one distinguishes two possible lines of action: On the one hand, researchers may either analyse small, defined sets of models, and classify or align them; as an example, Figure 7.3 shows how a dynamical cell cycle model could be compared with a simplified variant of the model from the same publication based on model files

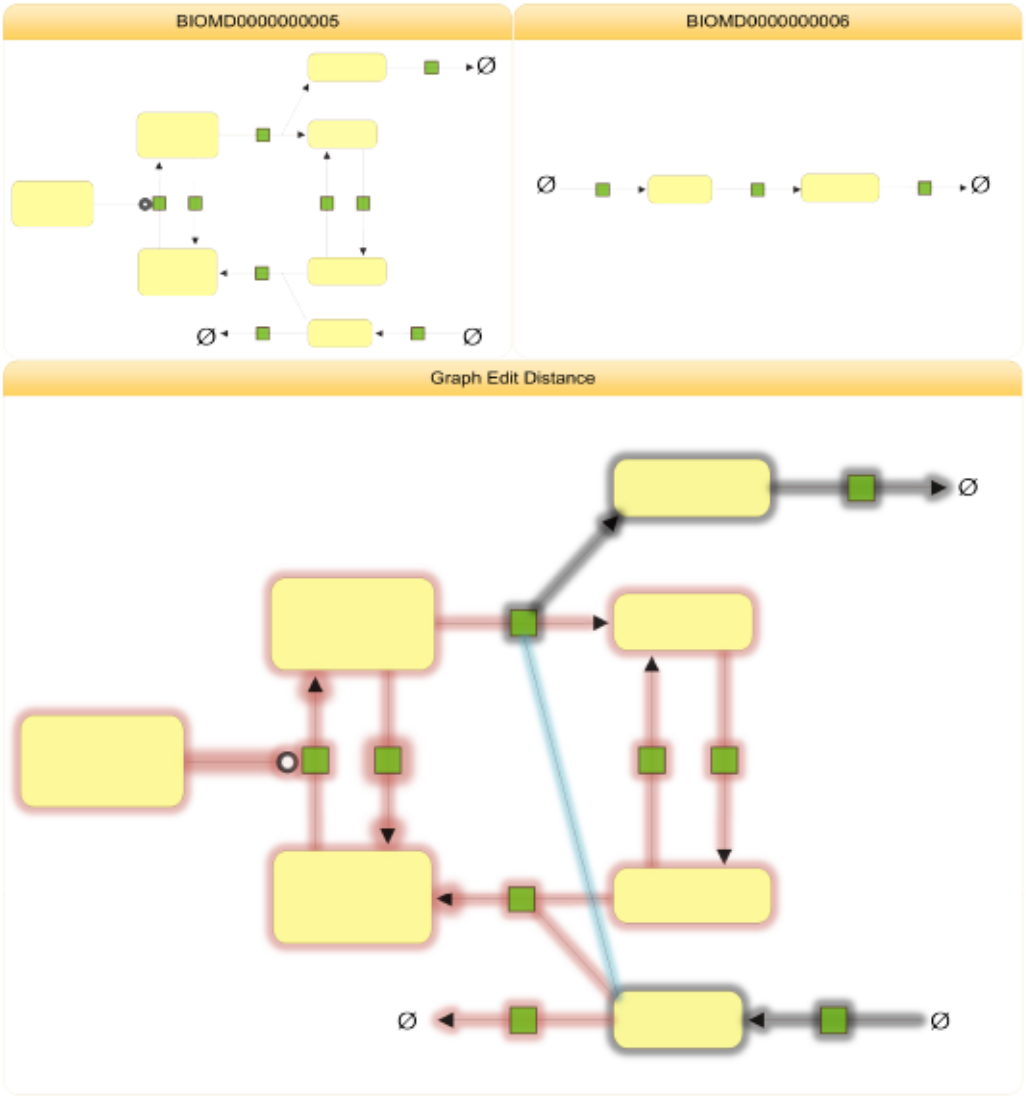


Figure 7.5.: *Graph edit distance.* To transform the network of BIOMD0000000005 into the network of BIOMD0000000006, 28 edit operations had to be performed; 16 edges and 11 nodes are deleted, 1 edge is inserted. Deletions are shown in red, insertions in blue and unchanged elements in grey. The node ∅, representing the empty set, is displayed multiple times for clarity.

available at BioModels. On the other hand, researchers may search for models in databases such as BioModels, possibly starting from some query model of interest. Methods for model comparison are equally important in both cases. The following describe basic use cases of model comparison and existing tools and methods devoted to these tasks.

7.5.1. Model search and clustering

Number and sizes of available models are increasing beyond what even the most well-read scholar can review or analyse, as exemplified in the growth of BioModels [Chelliah et al., 2015]. Probably the most common use of similarity measures is for model search, in which researchers query repositories to obtain models related to a given keyword, to a query model or to a data set.

In a keyword search, a user enters a set of terms to retrieve models that match these terms. In the simplest form, a model can be represented by a 'bag of terms', i.e. a list of relevant keywords or annotations, to which the user's query terms can be matched. A similarity between query terms and model can then be defined using measures from information retrieval. For example, a similarity score can be calculated from the frequency with which a term occurs, or from semantic similarity measures between terms [Henkel et al., 2015]. Instead of such 'bag of term' representations, models may also be represented in a structured form to incorporate network information, semantic annotations and other associated meta-data. Keyword search is state of the art in open model repositories. BioModels, for example, incorporates the aspects 'model encoding' and 'biological meaning' (see Table 7.1). A query by 'model encoding' matches exactly a set of terms associated with a model. A query by 'biological meaning' enables more sophisticated semantic searches. Search engines can be coupled with ranking algorithms to ensure that the most relevant search results appear in the top of the result list. The Physiome Model Repository, for example, uses a Lucene-based ranking algorithm that incorporates model encoding, biological meaning and other meta-data [Henkel et al., 2010].

If the starting point of a search is a model, then the aim of a search is to find similar models. SemanticSBML allows users to provide their own model as an input query. The search engine then finds SBML models that resemble the input model with respect to semantic annotations [Schulz et al., 2011]. The search operates on the openly available models from BioModels.

Similarity measures can also be used to calculate the similarity between models and query terms. Based on functions for model ranking [Henkel et al., 2010, Schulz et al., 2011], similarity measures can also help to cluster models into similar sets. For example, a cluster may organize models into thematic sets such as 'models describing metabolism', 'the cell cycle' or 'models showing calcium oscillations'. Thematic model sets may be characterized through semantic annotations [Alm et al., 2015] or through recurring structural patterns [Henkel et al., 2016b]. BioModels, for example, has implemented a web-based model browser that clusters models based on GO

terms. In the future, models may also be clustered based on biological motifs in their networks [Tyson and Novák, 2010]. Thematic sets can be searched and compared more easily, e.g. when constructing comprehensive models.

7.5.2. Network alignments

Network alignments provide a way to detect structural overlaps between pathways or networks. They are a basic tool in model merging and can be used to define similarity scores. For example, an alignment of kinetic pathway models with the metabolic network of yeast showed that large parts of the network are not yet covered, while central metabolism is heavily overrepresented in kinetic models [Schulz et al., 2011]. Another study showed how Boolean models can be coupled if the models adhere to certain modelling standards [Schlatter et al., 2011]. The resulting integrated model of cell–cell interaction between hepatocytes and Kupffer cells provides deeper insights into how different cell types respond to apoptosis or proliferation, and how IL-6 (Interleukin 6) and TNF (tumor necrosis factor) influence the interactions between cells of different types. Further recent examples of successful model integration include the prediction of phenotypes from genotypes using a whole-cell model [Karr et al., 2012] and the global reconstruction of human metabolism [Thiele et al., 2013].

These studies demonstrate how relative overlaps between networks can be quantified if the models contain a sufficient number of descriptive annotations to align the network components. However, if only few of the network components are annotated precisely, network alignment becomes a more difficult task: Only a few nodes can be matched based on the description of biological objects represented by network components, and the rest of the networks remain incomparable. One approach to address this problem is through semantic propagation [Schulz et al., 2012], a method that distributes semantic information in the network structure, either to infer missing annotations or to fully align the networks. The algorithm effectively gathers, in each model component, semantic information about the component's neighbours, its second neighbours, and so on. It then compares the models' components based on this neighbour information. In this context, neighbour elements are defined in an abstract sense. For example, reactions can be compared by annotations of their reactants, and cell compartments can be compared by the compounds they contain. Tests with blinded annotations have shown that model alignment and comparison can be strongly improved by using such propagated semantic information.

7.5.3. Model version control

New insights about a biological system may call for an adjustment of network structure, mathematical formulae or parameters of a model, resulting in new model versions. Another frequent reason for updates is error corrections. The comparison of model versions can help to keep track of the model's evolution in time, and it identifies points at which a model underwent major changes [Waltemath et al., 2013a]. BiVeS is a software library that aligns the XML encodings of two model versions, identifies and interprets changes and measures the changes' impact. BiVeS also considers characteristics of the model encoding format and differentiates between SBML and CellML. Single diffs are automatically annotated to terms of COMODI.

7.6. Discussion

Many efforts were made in the past years to improve the reusability of systems biology models and the reproducibility of associated results [Krause et al., 2011, Waltemath and Wolkenhauer, 2016, Waltemath et al., 2013b]. Model repositories collect curated models ready to be reused, provide semantic annotations and offer instructions on how to simulate the models appropriately. With standardized, annotated models being available, automatic model comparison has become a feasible task. The search for models is one important application of comparison.

Key challenges in model search include defining appropriate relevance scores for search results, defining measures for the quality of retrieved models and building frameworks for the access of a model's history. Such frameworks will enable researchers to follow a model's evolution. For example, modellers can learn how the knowledge about the cell cycle evolved. Similarity measures are also valuable for merging existing pathway models into larger cell models. The map of Human Metabolism developed in the ReconX project, for instance, relies strongly on previously published models [Thiele et al., 2013]. When two models are merged, parts of their networks may have to be exchanged or replaced by suitable alternatives, requiring tools that can compare both, models and model parts, and find the most similar matches. Because these tools need to solve similar problems and share similar difficulties, studying model similarity can be seen as a general task.

As a key challenge, the appropriate choice of a similarity measure needs to be identified. This leads to the following questions: How can we define computational measures that reflect a user's expectations about similarity? And how can these

measures, with various criteria, be applied in complex tasks such as search or merging? To create a general framework for model similarity and model comparison in future research, a number of issues need to be addressed.

Implement similarity measures for all model aspects. When comparing models, current software focuses on two of the aspects defined in this article, namely biological meaning and model encoding. Other model aspects are not yet commonly used. However, their implementation in existing algorithms is feasible and will improve model comparison. Dynamic behaviour, obtained from model simulations, could reveal similarities between biological processes during execution. These similarities will not show up when pathway structure alone is considered. One can directly compare simulated time series (as showcased by the Cardiac Physiology Web Lab [Cooper et al., 2016]) or build a system that compares their semantic annotations with quantitative or qualitative behaviour observed in simulations. A valuable resource of terms for dynamic behaviour is the TEDDY ontology. Likewise, improved similarity measures could be obtained by more extensively exploring graph matching and graph similarity algorithms [Berg and Lässig, 2004, Yan and Han, 2002], as suggested in [Rosenke and Waltemath, 2014]. Comparison of network structure can facilitate the matching of dynamical models to experimentally determined interaction networks. Equally, mathematical expressions could be directly compared to yield deeper insights into similarities of the models' behaviour. Furthermore, information that is attached to the model may become relevant, such as the purpose of an investigation or the modellers' intentions. However, this information must first be formalized—a new and interesting challenge.

Combine similarity measures. Today's software tools typically compare models by a single model aspect. As different similarity measures have proven useful for specific applications, one can expect that the combination of aspects would enable an even more powerful comparison of models, as in the following example: A scientist searches for a MAP kinase cascade model that contains regulatory feedback loops and shows dynamic oscillations. In a first step, a search tool could search for models of the specific biological system (using semantic comparison). Then, it could filter the intermediate results for specific network topologies. Next, a second filter could be applied to select models with oscillatory behaviour. Given sufficient additional information in the model repository, the remaining models could be compared based on their dynamic behaviour (e.g. by evaluating associated simulation descriptions in SED-ML format, and by comparing TEDDY terms therein). Eventually, an overall similarity measure joining the different aspects, could be defined. Such a

procedure requires suitable functions for combining the individual similarities in a single formula, and it should be possible for users to specify weights, i.e. a relative importance, for each of the aspects.

Support multiscale models. Future systems biology models may be much more complex than today's biochemical network models, ranging from spatial and stochastic cell simulations and modular whole-cell models to multiscale models of tissues, human physiology or populations of organisms. With different biological scopes and mathematical forms, new model aspects will become relevant (e.g. the way in which modules in a whole-cell model communicate with each other), and new similarity measures will be required. Some other aspects (e.g. the lists of molecular species described, or the occurrence of dynamic oscillations), known from current network models, will still hold. A particular challenge will be posed by models that are internally and hierarchically structured, e.g. body models composed of organ models, which are composed of cell models, and so on. In these cases, similarities may first be defined on the level of individual modules, and then be propagated up to the level of entire models. One can expect that software support for model comparison across multiple levels will become a requirement for flexible, semi-automatic model construction

Improve software support. To allow for extended similarity measures and to integrate them easily into software applications, new tools need to be developed. Interoperability through standard formats and common software libraries must be ensured. Such tools could incorporate a large set of existing similarity measures and provide functions for projecting models onto their relevant aspects. In addition, the scientific community has developed shared resources for models and associated metadata, widely accessible through graph databases [Henkel et al., 2015] or publicly accessible Semantic Web resources [Jupp et al., 2014]. Tools to process models and determine similarity should be able to access and interoperate with these shared resources.

Provide tools to align models and data by similarity. This chapter already discussed how similarity measures can link models to other models or to user queries. It is equally interesting to investigate how the process of linking models and experimental data can be improved. One can envision that models and other data can be compared by projecting them onto a common aspect. For example, models and a patient's cancer genomics data set can be projected to proteins appearing in both, model and data set. Subsequently, semantic annotations of biological processes can be compared between the model entities and the data items. Afterwards, similarity

measures for biological meaning can help identify whether observations in the patient match a particular state predicted by a model.

Develop intuitive user interfaces. When offering purpose-driven similarity measures, it is important to communicate the details of the scores to the users. Consequently, there is a need to develop clear and intuitive user interfaces that show both, the results of a similarity score and the details of calculation. For example, a system for model retrieval should return a ranked list of models, with a detailed description of filtering processes and relationships between models and query. Some software tools already visualize element alignments between models as network graphs (e.g. BudHat [Waltemath et al., 2013a], SemanticSBML, STON [Touré et al., 2016]), or present ranking scores for retrieved models (e.g. MASYMOS [Henkel et al., 2015], SemanticSBML). However, an explanation of the steps leading to the similarity scores will increase the trust of the users clarify which of the search results are most relevant.

7.7. Conclusions

Similarity between models is assessed by a variety of software applications. This chapter provided a systematic classification and a review of model similarity measures. It further introduced and discussed aspects that help with determining the similarity between models: the model encoding, when comparing versions of a model; the mathematical description of a model, when investigating the systems' dynamic behaviour; the biological elements appearing in a model, when searching for models of a specific biological system or phenomenon; the network structure, when investigating the reuse of models as submodels in large networks; the parameter values in a model, during functional curation; and simulation outcomes, when comparing behaviour and sensitivity of a model. Consequently, a general framework for model similarity, based on a systematic treatment of models and their aspects is desirable. Such a framework will enhance the automated processing of models and may have numerous applications in computational systems biology.

If your contribution has been vital there will always be somebody to pick up where you left off, and that will be your claim to immortality.

— *Walter Gropius*

8. Habitat status report

The research described in the previous four chapters spans a period of 12 years. This chapter provides an update on the research described and shows the further developments in model retrieval (see Section 8.1), version control (see 8.2) and analysis (see 8.3). It also briefly explains a similar approach of using a graph database to store and query biological pathway data (see 8.4). Finally, the transition of concepts developed for the model in the systems biology domain into the medical informatics domain is described sing the example of CovidGraph (see 8.5).

8.1. Development of model retrieval and ranking

When Henkel et al. [2010] (see Chapter 4) was published, state-of-the-art model retrieval was mostly done by database queries and ontology filtering [Li et al., 2009] or out-of-the-box content management system lookup [Yu et al., 2011]. With the rising numbers of available models (see Figure 8.1 and in addition an increasing complexity and size of models, simple database queries and content management system search capabilities hit a limitation. More precisely, providing a user with a large number of matching but unranked results was not acceptable and hampered model reuse and reproducibility - evermore so with the publication of over 143070 semi-automatically generated pathway models [Büchel et al., 2013]. Furthermore, considering model semantics and accompanying data for model retrieval became more important. Next to Henkel et al. [2010], similar approaches for model retrieval and ranking were developed. For example, Schulz et al. [2011] developed an algorithm to align models by their semantic annotation and Wimalaratne et al. [2014] provided semantic access to BioModels via a SPARQL endpoint.

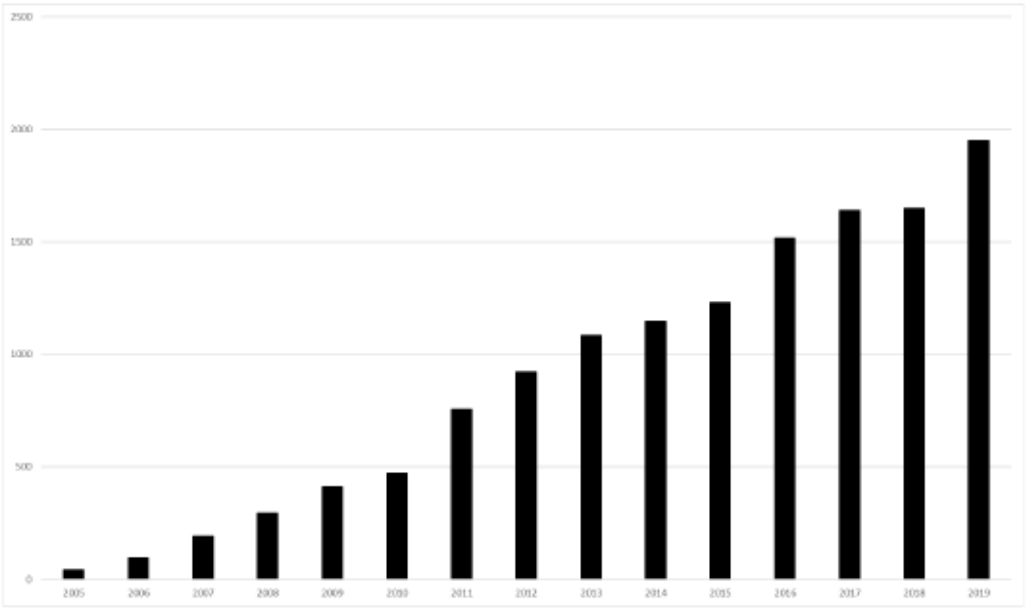


Figure 8.1.: When Henkel et al. [2010] was published, BioModels contained approximately 500 models. In 2022 the database contains almost 2426 models. However, BioModels changed its model publication scheme to a continuous publication and discontinued the provision of database snapshots, making it unfeasible to display the model numbers on a yearly basis.

With the development of the graph database back-end (MaSyMoS, see 6), the retrieval an ranking capabilities were expanded to also include (next to SBML) CellML, SED-ML and a selection of accompanying ontologies. While the original approach of ranking and retrieval was prototypically implemented in BioModels in 2009, the extended approach was implemented in the Physiome Model Repository in 2013. In the following, changes made to the index building and model retrieval after the publication of Henkel et al. [2010] are described.

8.1.1. Model Ranked Retrieval Engine

The idea of model retrieval and ranking as described in Chapter 4, based on [Henkel et al., 2010], works on the premise of being a stand-alone index without a database back-end. As described in Chapter 6, based on [Henkel et al., 2015], this premise changed to also incorporate model structure and accompanying data into a model search. Consequently, the model retrieval and ranking engine (MoRRE) had to be adapted to efficiently use the graph database back-end (MaSyMoS).

8.1.2. Index building

The original retrieval and ranking idea is model based. Every accessable information encoded in a model is stored in a model index. The semantic index holds all model annotations, represented by an URI, enriched with the textual descriptions behind that URI and a link to the model where the annotation is used.

With MaSyMoS the indexing is done more explicit, following the idea of node- and relationship-based indexing. For each type of node holding information valuable for searching (i.e., model, sedml, annotation, publication or person), a single index holding node type specific fields, analyzer and weights is created. Figure 8.2 provides a more detailed view on the 5 indices created. For example, the annotation index

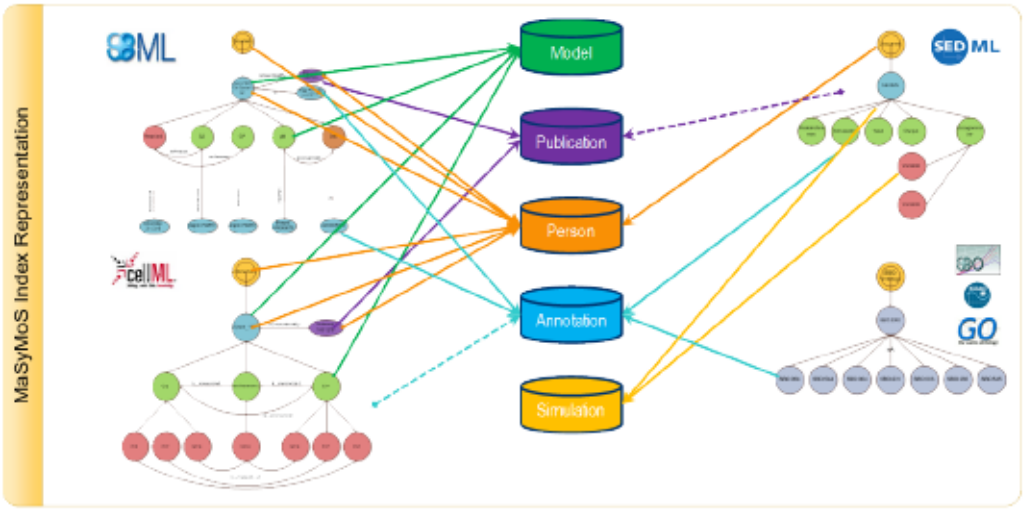


Figure 8.2.: Using the model Tyson1991 - Cell Cycle 6 var [Tyson, 1991], this Figure shows the connections between SBML, CellML and SED-ML entities as encoded in MaSyMoS and the five indexes created. The annotation index, for example, contains annotation information attached to the SMBL model node, SBML species, compartment and reaction nodes, the SED-ML Simulation node. Also connected are terms from stored ontologies, e.g. the referencing Kinetic Simulation Algorithm Ontology term used in the SED-ML simulation). A dashed line denotes missing structured information in the corresponding sources, although this information could, in general, be added to the index.

holds the following fields:

URI A Uniform Resource Identifier unambiguously identifying an ontology entry. This information stems from a resource node which is attached to a model, a model entity, a simulation description or a simulation description entity (e.g., GO:0005892, acetylcholine-gated channel complex). A keyword analyzer is used for this field, meaning the URI is stored in the index exactly as provided

by the resource node. The attached weight is 4, as retrieving an annotation by searching for an exact URI as keyword is deemed as significant.

ResourceText is a textual representation of content behind a URI. It is created by resolving the URI provided by the resource node into a URL and loading the textual representation as HTML from the web. The standard analyzer strips unnecessary HTML encoding, stop words (e.g. and, or, in, ...), removes punctuation marks and special chars and transforms the remaining text to lowercase. The result is only stored in the index but not in the resource node. The attached weight is 3, as the textual description of an annotation is deemed less significant compared to the URI.

NonRDF The specification of SBML and CellML allow annotations to be free text (more specifically, the requirement for an annotation is the XSD "any" element). Consequently, if no URI can be identified in the resource node, the standard analyzer is used to process the provided unstructured information. In this case the URI and ResourceText fields are left blank. The attached weight is 2, as this kind of annotation is mostly used by tools to provide layout information.

NodeID This field holds the internal reference to the resource node and can not be used for retrieval.

8.1.3. Model retrieval and ranking

Each of the 5 described indices is independent and can work on its own. Subsequently, a user is able to retrieve nodes representing a model (e.g., by the BiomodelID), a simulation description, a publication, a person or an annotation. However, it can be assumed that is user is usually interested in retrieving a model. MaSyMos, as described in Chapter 6, connects models with information regarding publication, annotation, simulation description and others in a structured graph. Thus, if a query retrieves, for example, a publication node it is easy to traverse the graph to also retrieve all model nodes, respectively models, connected to this publication node. The same holds for annotation, person and sedml nodes (if they are connected to a model via their modelreference node) whereas the model index retrieves model nodes directly. Figure 8.3 shows an example retrieval describing this process.

For each node retrieved from an index by a query, a similarity score is assigned[1].

[1]Score is based on the vector space model and uses normalized term frequency, inverse document frequency, number of matching query terms per document.

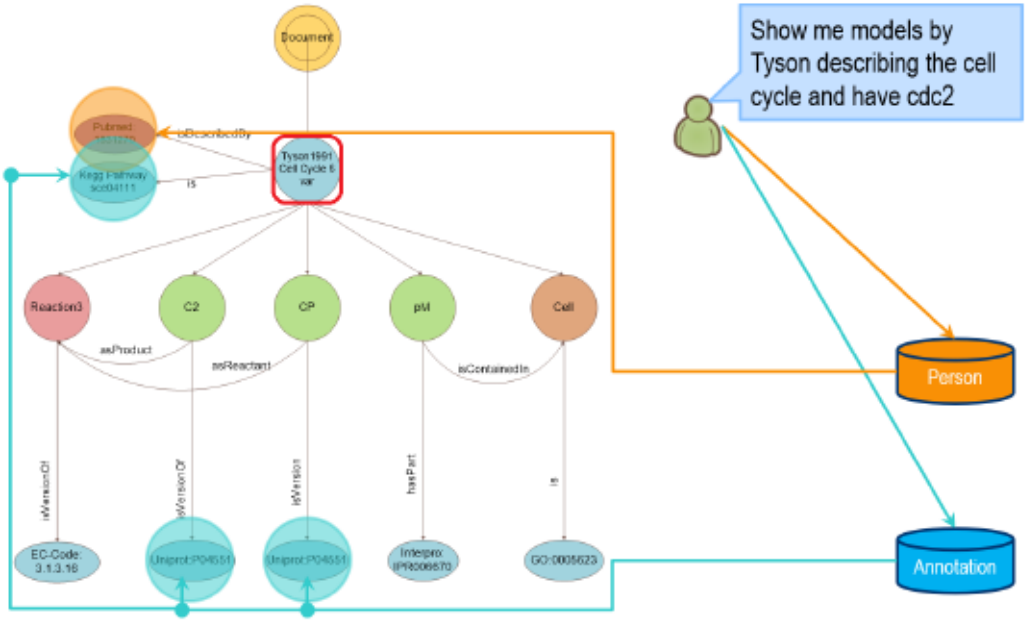


Figure 8.3.: Using the model Tyson1991 - Cell Cycle 6 var [Tyson, 1991], this Figure shows the retrieval process. The user query has the keywords Tyson, cell cycle and cdc2. The keywords are used to query different indices, for simplification only person and annotation index are shown here. For example, the annotation index retrieves three annotation nodes (blue circle), belonging to a model node (directly or via species nodes (green)). Consequently, the model node (red square) is put in the list of relevant nodes for this query.

For the example shown in Figure 8.3, 3 annotation nodes are retrieved from the annotation index - each node holding a score. In addition, 1 node form the person index is retrieved, also holding a score. All 4 nodes are traversed up to the model node and the score is passed on accordingly. Subsequently, the list of relevant model nodes now contains 4 times the same model but with different scores. To compute a valid ranking, a post processing is done by collating identical model nodes. The score for the collated list of model nodes is computed by summing up the scores of each identical model node retrieved and divided by the maximum number of retrieved nodes related to a model node[2] This is a computational fast way to compute a reasonable model ranking from scores of retrieved nodes related to a model. However, to be able to compare results from different queries a rank aggregation is desirable [Nassar et al., 2015].

[2] $sc(M^i) = \frac{\sum_{h=1}^{n} sc(P_h^i)}{max_n(M^0,\ldots,M^i,\ldots,M^m)}$ with $sc(M^i)$ as the score for an entry of a collated model list, $sc(P_h^i)$ as the score of a node related to model node i, $max_n(M^0,\ldots)$ is a function returning the highest number of retrieved nodes related to a model.

8.1.4. Rank aggregation

Given a set of rankings $R_1, R_2, \ldots, R_m$ of a set of objects $X_1, X_2, \ldots, X_n$ the task is to produce a single ranking R that is in agreement with the existing rankings. This task can be solved by methods either using the position or the score of an object.

As aforementioned, when searching different indices using a query q, a ranked list of results (nodes) will be generated for each index. By transforming the nodes into model nodes (traversal) and collate the results, a ranked list of scored model nodes is created per index. These lists can be viewed as a set of input rankers $R_{in} = \{R^1, ..., R^s\}$. The initial aggregate ranker R^A is created as described above by adding up the scores of each model from each ranker and normalize them by computing the highest possible frequency of models in the rankers and dividing the scores by the highest frequency. To compute a rank aggregation 4 methods are chosen:

Modified Adjacent Pairs: Given a set of input rankers $\{R^1, ..., R^s\}$ and the initial aggregate ranker R^A, a local optimization to R^A is computed by swapping adjacent models. The aim is to improve the average Kendall-Tau ([Adali et al., 2007]) distance between R^A and $\{R^1, ..., R^s\}$.

CombMNZ: A score-based method to aggregate rankings described in [Adali et al., 2007]. Given a set of input rankers $\{R^1, ..., R^s\}$ and an initial aggregate ranker R^A, the algorithm first computes the normalized Borda rank of each model and then multiplies this value by the number of model occurrences in the different input rankers. The aggregate score for each model is computed. The set of aggregate scores is then sorted in descending order.

Local Kemenization: Given a set of input rankers $\{R^1, ..., R^s\}$ and an initial aggregate ranker R^A, a locally Kemeny optimal aggregated ranker is created ([Dwork et al., 2001]). For each model M^i in R^A and for each of the higher-ranked models in R^A, M^j, a test if the majority of rankers in R_{in} rank M^i better than M^j is conducted. If so, M^i and M^j are swapped.

Supervised Local Kemenization: A weighted version of local Kemenization as described by [Pujari and Kanawati, 2012]. Following the idea that not every ranker is equally important, Supervised Local Kemenization assigns weights on input rankers $\{R^1, ..., R^s\}$. Consequently, an aggregated result is influenced by the initial rankings of the model, and also by the weight of the corresponding ranker.

Computing a rank aggregation is computationally harder than the default approach of normalizing the scores by the maximum of retrieved nodes. However, rank aggregation algorithms provide a smoother transition from individual ranking scores to an aggregated ranking. For example, local kemenization considers only the rank and not the score of each retrieved node in an aggregate ranker. Hereby, the rank aggregation cancels out outliers that were only retrieved by one ranker but with a comparable high score.

Rank aggregation is also useful when different versions of one model are stored (see Subsection 8.2.4). Here the question arises how to score a particular model, in case more then one version of this model is retrieved and the retrieved versions return different scores. Rank aggregations can be used to determine which of the retrieved model versions should be presented to the user.

8.2. Proceedings in model version control and provenance

The paper Waltemath et al. [2013a] (see Chapter 5) defines conceptual requirements for model version control and outlines methods for identification and justification of changes. This Section briefly describes the further algorithm development and refinement [Scharm et al., 2015], the COMODI ontology [Scharm et al., 2016] to annotate changes in a model and explains how to keep track on the evolution of computational models in a database [Scharm et al., 2018].

8.2.1. A refined algorithm to detect differences in models

The prototypical implementation for the first model version control application [Waltemath et al., 2013a] was based on the XYDiff algorithm [Cobena et al., 2002] which is suited to compare generic XML documents. For model version control, the schemas for SBML and CellML are well defined. Consequently, [Scharm et al., 2015] refines the approach and describes an algorithm tailored towards SBML and CellML difference detection. Figure 8.4 outlines the steps for such a comparison that explicitly takes into consideration a models' encoding, the structure of biological networks and mathematical expressions.

The first step is to pre-processes the model documents, translating them into 2 internal tree structures (T_1 and T_2 and assigning signature hash sums σ and weights ω to each node. The signature hash denotes the a node and its subtree, it is used

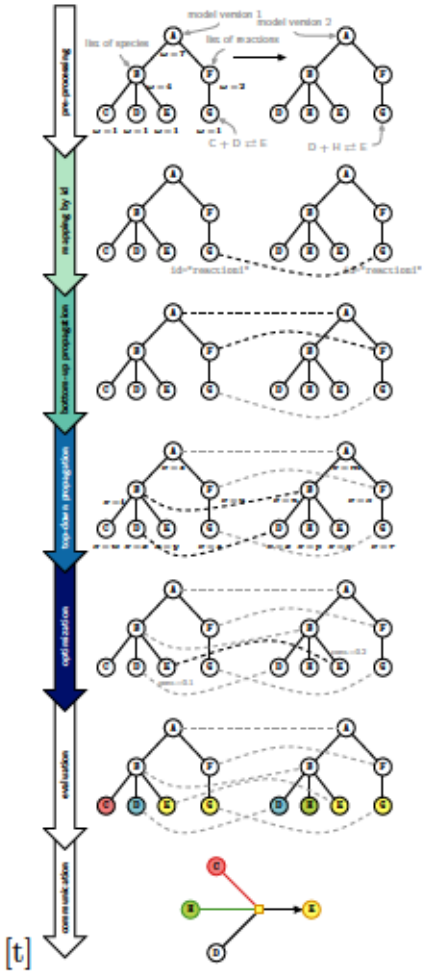
to determine changes in the tree structure. The weight as a function based on the length of text stored in a node. Subsequently, a parent node has always a weight higher than the weight of its children.

The next step is an ID-based mapping. In contrast to the original algorithms which uses the XML id-attribute, the refined algorithm computes a mapping based on the biological identifiers as they are deemed more significant (the XML id-attribute is often created automatically by modelling tools).

The step of bottom-up propagation takes care of the upward propagation of mapped nodes in T_1 by evaluating each nodes' children with a depth-first traversal of T_2. The potential connection of nodes are scored and the algorithm selects the best matching candidates.

The top-down propagation, as the next step, matches nodes of T_2 on nodes of T_1 by hash signature. In the following nodes of T_2 are ordered descending by weight into a priority queue, starting with the root node only. For the node with the biggest weight (meaning the biggest subtree) a set of mapping candidates is collected. Preferred candidates already have a mapping between ancestor nodes. If no mapping can be established, all children of the current node are added to the queue, the next iteration starts.

The last step is an optimization of the results computed so far. For every mapping, the unmatched children are compared and to find missing mappings.



[t]

Figure 8.4.: This schematic of the mapping procedure depicts the steps necessary for model comparison and difference detection. Using two constructed example models (having the Nodes A–H) the mapping, bottom-up and top-down processing, and optimization is explained graphically. Dashed lines indicate mappings between the nodes, whereas σ and ω are a node's signature and weight, respectively. A node's colour correspond to the detected modification: updates are yellow, inserts are green and deletes are red, and moves are blue.
This Figure was originally published in Scharm et al. [2015] (as Figure 2)

Here, a distance matrix is computed by calculating a ratio of matching and missing attributes of each compared child node.

Lastly, a language specific post-processing is applied to the list of matching nodes. This step removes computed matches that are not reasonable given the SBML or CellML specifications (e.g. moving a `listOfModifiers` from one SBML reaction to a different one is not reasonable). Finally, using the trees T_1 and T_2 in addition to the computed matches, a type (update, insert, update, delete) is defined. For example, if an entity is present in T_2 but not in T_1 the type of change is an insert.

Finally, the adjusted mapping outcome can be exported in both machine and human readable formats.

8.2.2. Characterising differences in versions of computational models

Waltemath et al. [2013a] argued for model changes to be classified and justified in order to allow model provenance and evolution tracking over time. The COMODI ontology (COmputational MOdels DIffer) described in Scharm et al. [2016] aims to provide terms for such a classification and justification. COMODI can be used to describe and annotate changes in models. Such semantically enriched model changes help to understand and reuse a model.

Changes in a model can be diverse (e.g., error correction, new knowledge gained, new exchange format specification or model curation) and occur on multiple semantic levels. For example, a change in a model can impact a models reaction network, parameters, annotations or entity names. Depending on the kind of change a model may produce different results when simulated (e.g., change in reaction network or parameters). On the other hand, a model's visualization might change if an annotation from the Systems Biology Ontology (SBO) is altered (SBO annotations are used to determine glyphes for model visualization with SBGN).

To create COMODI a selected set of models from BioModels (SBML) and Physiome Model Repository (CellML) was manually analyzed. Afterwards a complete set of model changes were calcualted using the BiVeS tool [Scharm et al., 2015, Waltemath et al., 2013a]. The insights gained by this analysis were used to draft candidates for the ontology. In the following the candidates were manually aggregated and classified. Afterwards, concepts from the two used standard exchange formats SBML and CellML, and terminology from the XML domain was added.

COMODI contains four major concepts connected to the major concept `Change`:

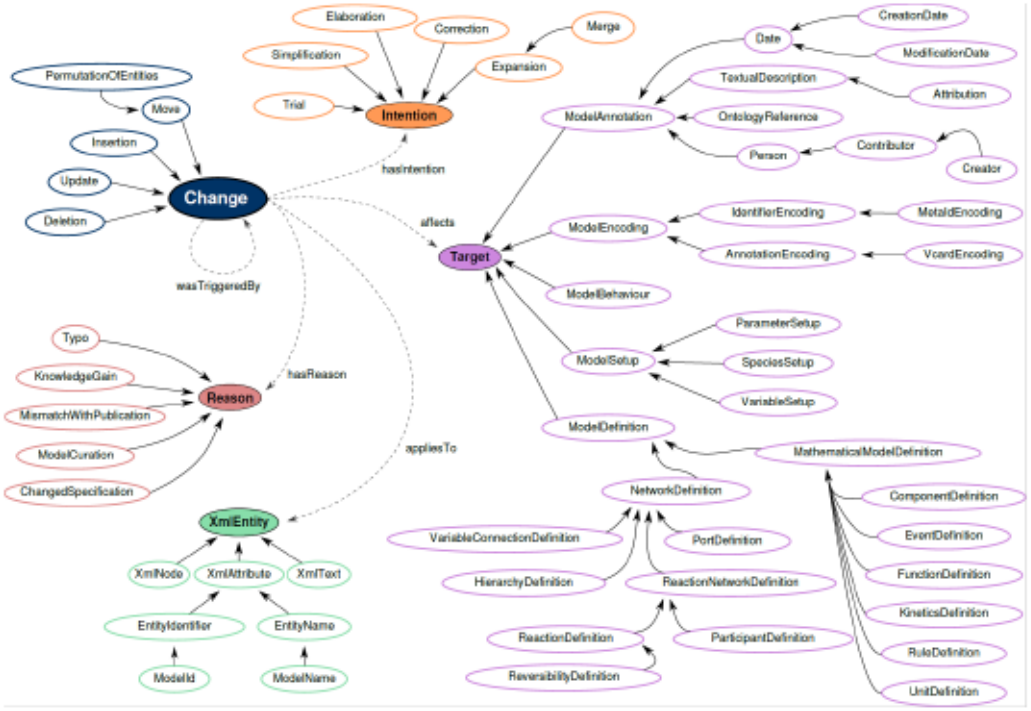


Figure 8.5.: Structure of the COMODI ontology as developed by Scharm et al. [2016] is used to annotate differences between models. The major concepts are `Change`, `XmlEntity`, `Intention`, `Reason`, `Target`. Each major concept is further refined into more specific sub-concepts and used to annotate different aspects of a model change.
This Figure was originally published in Scharm et al. [2016] (as Figure 2)

`XmlEntity`, `Intention`, `Reason`, `Target` as shown in Figure 8.5. A model `Change` can be classified as a `Move`, `Insert`, `Update` or `Delete`. The concept `XmlEntity` further defines if an `XmlNode`, `XmlAttribute`, or `XmlText` element was changed. The purpose of a change is encoded by the concepts `Intention` (providing the aim of a change) and `Reason` (explaining the cause of a change). The concept `Target` reasons about the aforementioned impact of a model change and is further defined by Scharm et al. [2016] as:

> "The `ModelEncoding` corresponds to the formal encoding of the model document. Terms of this branch can, for example, be used to describe an update of the underlying SBML specification.
>
> The `ModelAnnotation` branch corresponds to the semantic layer of a model document. Terms of this branch can, for example, be used to capture changes in the annotations.
>
> The `ModelDefinition` refers to the actual biological system, for example a reaction network. Terms of this branch can, for example, be used to specify the parts of a model that are affected by a change.

> The `ModelSetup` branch can be used to describe changes in the simulation environment. Terms of this branch can, for example, be used to describe changes in parameter values.
>
> The `ModelBehaviour` links to the TEDDY ontology [Courtot et al., 2011]. Thus, it is possible to capture changes in the dynamics of the system. Such changes may, for example, affect the stability characteristics."

8.2.3. Evolution of computational models

Scharm et al. [2018] states that "The reuse of models is still impeded by a lack of trust and documentation. A detailed and transparent documentation of all aspects of the model, including its provenance, will improve this situation" - an key demand also backed by Waltemath et al. [2013a]. It was already stated that small, undocumented changes to a model can have a major impact on the simulation result or lead to failure of the model being simulated, respectively.

To make a statement about the quantity, identity and possible impact of model changes, Scharm et al. [2018] analyzed 13734 models from BioModels Database and the Physiome Model Repository, more precisely the differences between the corresponding model versions. BiVeS [Scharm et al., 2015, Waltemath et al., 2013a] was used to identify changes between two successive model versions, the data gained was analyzed using R-scripts and subsequently visualized[3] and interpreted. For example, one finding is that models are changed approximately once a year in the first five years of publication (an average of 4.49 versions per model).

8.2.4. Storage solution for models with multiple versions

In his Bachelor Thesis [Peters, 2016] conceptualizes and implements the idea of storing multiple versions of biological models in the MaSyMoS database, hereby building a bridge between model storage and retrieval, and model provenance by version control. To achieve this goal, Peters [2016] uses the BiVeS tool [Scharm et al., 2015, Waltemath et al., 2013a] to compute model changes and annotate them using terms from the COMODI ontology [Scharm et al., 2016]. Subsequently, the MaSyMoS storage concept (see Chapter 6 is extended to store the computed model changes and link them to COMODI annotations and the relevant model versions (see Figure 8.6, respectively.

[3]ModelStats website: `https://most.bio.informatik.uni-rostock.de/`

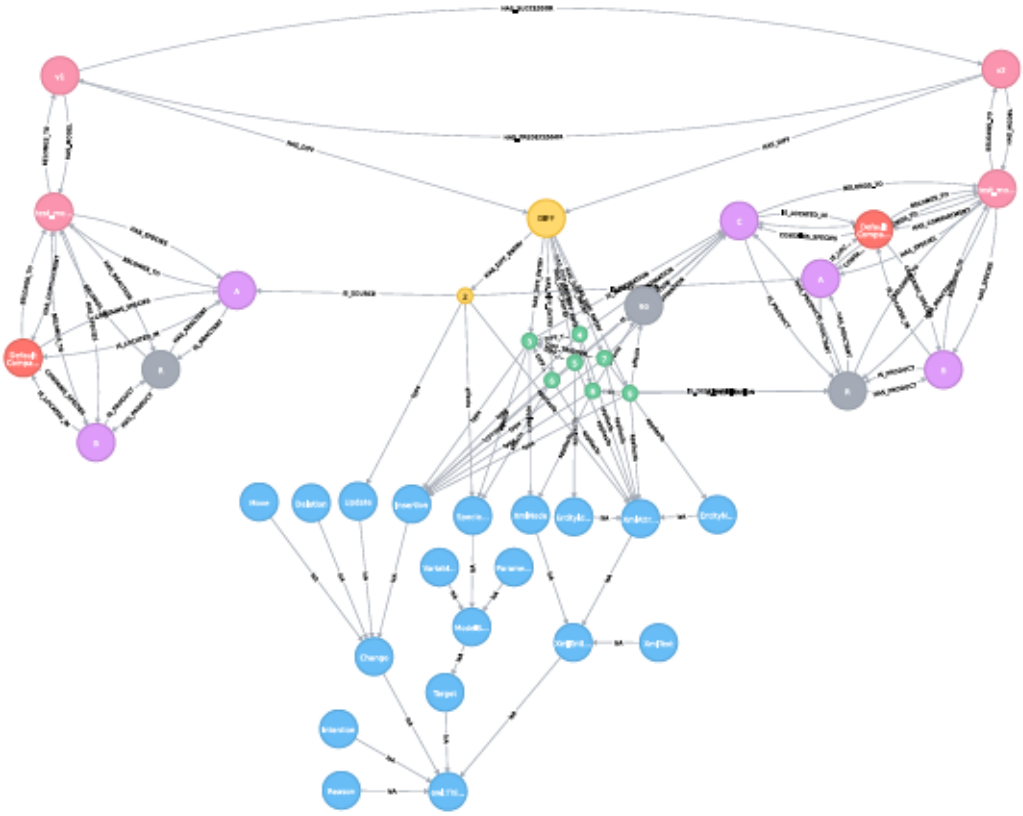


Figure 8.6.: This Figure shows a graph representation of a delta (center, yellow and green nodes) between two toy models (left and right, red and purple nodes). Each computed difference is linked to a term from the COMODI ontology (bottom, blue nodes).
This Figure was originally published in Peters [2016] (as Figure 5.1)

The extended storage concept allows to access, query and compare different version of a model on the database level and integrates semantic annotation of changes between model versions. The semantic annotations can be used to compare and classify model changes on a more abstract level.

8.3. Data analysis using MaSyMoS

With MaSyMoS (see Chapter 6), storing and linking models, simulation descriptions, ontologies and annotations in one database, and MoRRE (see Section 8.1.1) adding enhance retrieval and ranking mechanisms, the stored data is available for analysis. In the following two examples of analysing models and model annotation are presented. Both examples show the value of MaSyMoS for data and model analysis

8.3.1. Annotation-based feature extraction

Model repositories such as BioModels Database provide computational models of biological systems for the scientific community. These models contain rich semantic annotations that link model entities to concepts in well-established ontologies such as the aforementioned Gene Ontology (GO) or Systems Biology Ontology (SBO). Consequently, thematically similar models are likely to share similar annotations.

The first analysis example is based on model annotation and how this semantic annotation can be used to characterize sets of models. These characteristics may improve model classification, allow to identify additional features for model retrieval tasks, and enable the comparison of sets of models.

Alm et al. [2015] discusses four methods for annotation-based feature extraction from model sets and testes all methods on sets of models in SBML format which were composed from BioModels. To characterize each of these sets, extracted concepts (ontology terms used in model annotation) from three frequently used ontologies, namely Gene Ontology, ChEBI and SBO are analyzed. The four methods take into account the information content of a term used for annotation and the frequency a term is used as an annotation. The data necessary for these computations is provided by MaSyMoS. Three of the four proposed methods are deemed to be suitable to determine characteristic features for arbitrary sets of models.

One key finding is, that the selected features vary depending on the underlying model set, and at the same time are also specific to the chosen model set. Thus, it is possible to derive a similarity value to compare model sets. Alm et al. [2015] also shows that the identified features map on concepts that are higher up in the hierarchy of the ontologies than the concepts used for model annotations. The analysis also reveals that the information content of concepts in ontologies and their usage for model annotation do not correlate.

In addition, an analysis of concept vs. annotation distribution in SBO is presented (see Figure 8.7). It becomes obvious that the concepts are unequally distributed across the seven top-level branches (Figure 8.7, top). This is explained by the design of the SBO and its orthogonal branches. For example, the branch modeling framework (SBO:0000004) lists a "set of assumptions that underlay a mathematical description" whereas the branch mathematical expression (SBO:0000064) contains "formal representation of a calculus linking parameters and variables of a model". Consequently, one expects more entries for mathematical expression than for modeling framework. In conjunction with the application of SBO in model annotation, concepts of some branches are annotated more frequently (Figure 8.7, bottom).

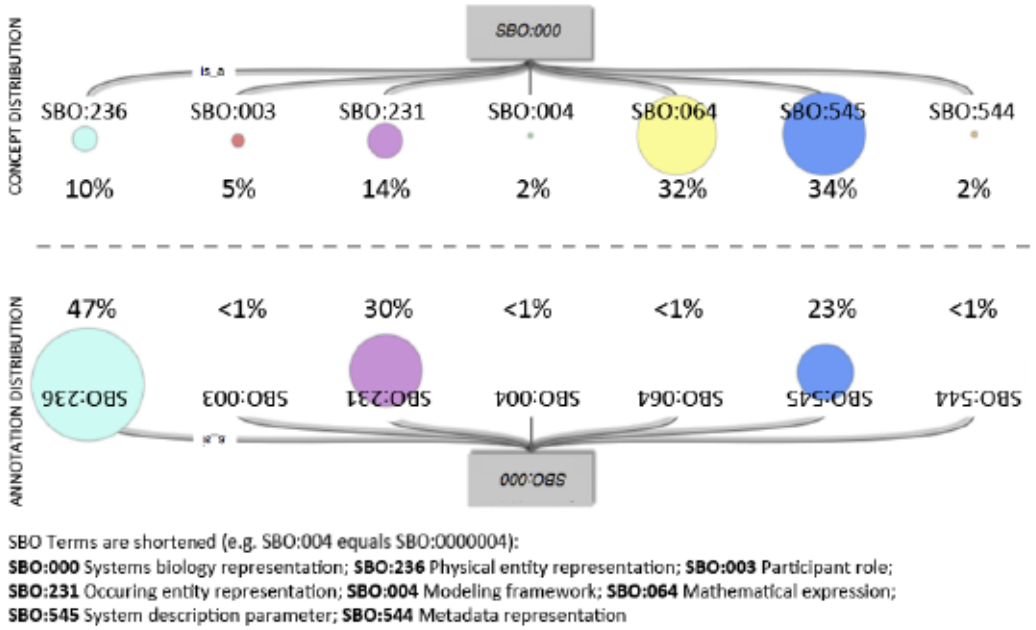


Figure 8.7.: Overview of the concept distribution in the seven branches of the Systems Biology Ontology (SBO). The size of the colored circles visualizes the number of concepts summarized by each branch. The bottom mirrored image visualizes the distribution of annotations from all models in the BioModels Database test set.

For example, the branch physical entity representation (SBO:0000236), which is a "representation of an entity that may participate in an interaction, a process or relationship of significance", contains only 10% of SBO concepts, but 47% of the model annotations link to that branch. One would expect that the characteristic features follow the distribution of the model annotations as seen in the lower part of the figure. Indeed, when applying a method that takes into account information content and entity frequency, the selected SBO features show a distribution (66.6% physical entity representation (SBO:0000236), 6.6% participant role (SBO:00000003), 13.3% occurring entity representation (SBO:0000231), 6.6% mathematical expression (SBO:0000064), and 6.6% systems description parameter (SBO:0000545)) that almost reflects the distribution in Figure 8.7 (bottom).

8.3.2. Pattern recognition

Computational models in biology encode molecular and cellular processes. Many of these models can be represented as biochemical reaction networks. Studying such networks, one is mostly interested in systems that share similar reactions and mechanisms. Typical goals of an investigation thus include understanding of model parts, identification of reoccurring patterns and recognition of biologically relevant

motifs. Specifically for the problem of finding patterns in large networks only partial solutions exist.

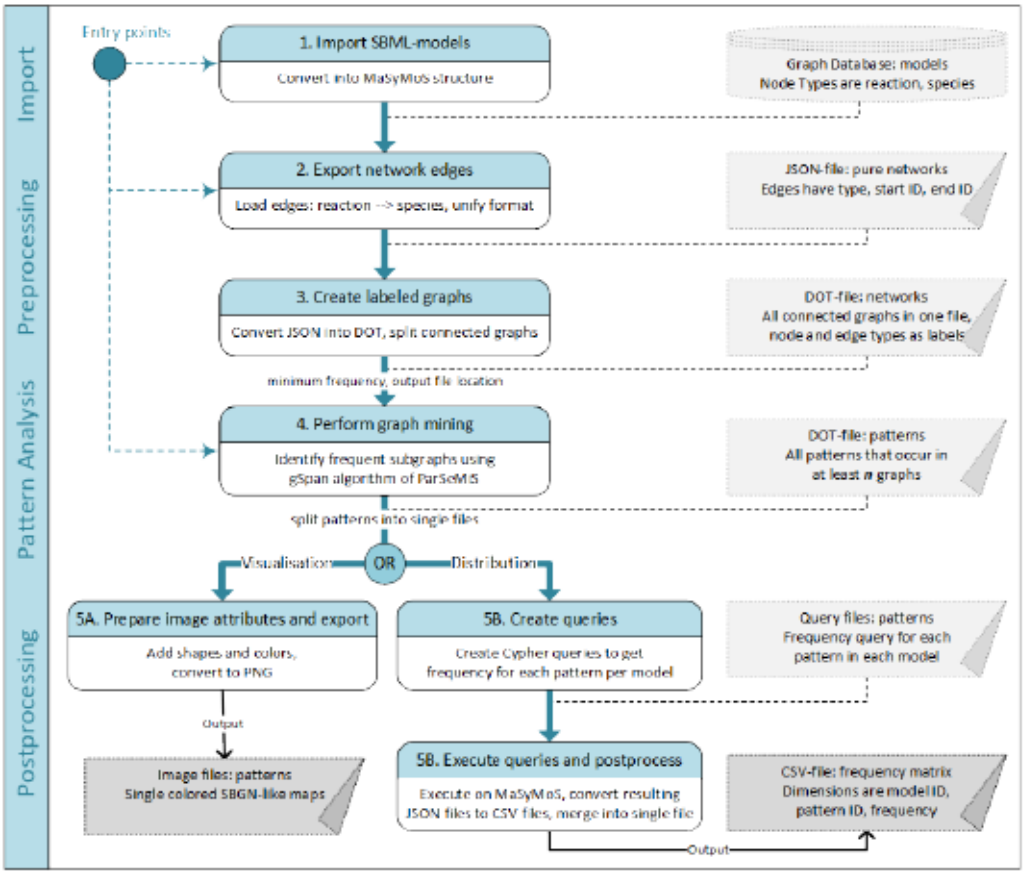


Figure 8.8.: Workflow as proposed in Lambusch et al. [2018]. Numbered, oval boxes describe workflow steps. Rectangular boxes describe data produced by each step and taken as an input for the following step. Starting at multiple entry points is possible depending on the available data and format. The first step is to import models into the MaSyMoS graph database. From there the reaction networks are extracted and converted to a uniform structure. The converted reaction networks are used as an input for the subgraph mining step to identify patterns. Subsequently, two options to further process the pattern descriptions are available. On the one hand, images representing the patterns can be generated (output on the bottom left). On the other hand, patterns can be fed back to the database to create a feature matrix showing the distribution of identified patterns among the models (output on the bottom right).

In contrary to the analysis above, the second example entirely focuses on the reaction networks that are encoded in models. Lambusch et al. [2018] proposes a workflow that identifies frequent structural patterns in biochemical reaction networks (encoded in SBML). The workflow (see Section 8.8) utilizes a subgraph mining algorithm to detect the network patterns. The data necessary for this pattern detection is provided by MaSyMoS. Once patterns are identified, the textual pattern description can

automatically be converted into a graphical representation. Furthermore, information about the distribution of patterns among a selected set of models can be retrieved by querying MaSyMoS for occurrences of the pattern of interest.

As biologists have an interest in classifying models by their function, Lambusch et al. [2018] makes a step towards motif identification by pattern recognition. Tyson and Novák [2010], for example, were interested in the mechanisms of information processing. They showed that complex networks could be decomposed into simple patterns, each fulfilling specific functions within a cell. These patterns were postulated as common motifs in biochemical reaction networks. It remains an open question how and how frequently these motifs are encoded in a model, however pattern recognition enables to compare patterns found to be encoded in models reaction networks and motifs postulated by Tyson and Novák [2010]. The occurrences of frequent patterns may give insight into the encoding of central biological processes, evaluate postulated biological motifs or serve as a similarity measure for models that share common structures.

The workflow was validated with 575 models from the curated branch of BioModels (Release 29). Lambusch et al. [2018] highlights interesting and frequent structural patterns and provides exemplary patterns that incorporate terms from the Systems Biology Ontology. The proposed workflow can be applied to a custom set of models or to models already existing in an instance of MaSyMoS.

8.4. STON

As MaSyMoS, STON (SBGN TO Neo4j, [Touré et al., 2016]) is based on the idea of efficiently storing biological information encoded in an exchange format, in this chase SBGN (Systems Biology Graphical Notation) [Le Novère et al., 2009], in a graph database to enable better processing of the encoded biological relationships and support queries on the structure of biological networks.

STON imports and translates metabolic, signalling and gene regulatory pathways represented in the Systems Biology Graphical Notation into a graph-oriented format compatible with the Neo4j graph database, the same graph database engine that is used for MaSyMoS. The representation of biological pathways in graph databases requires translation rules. The SBGN-ML format already describes a graph-like structure: Glyph nodes (representing entities) interact with each other by means of glyph arcs (representing relations). The translation process has been facilitated by the similarity of SBGN-ML and Neo4j. SBGN glyph entities are nodes in Neo4j, and

SBGN arcs are relationships. Additional information (e.g., ID, state variable, unit of information.) is retrieved as properties for Neo4j nodes and relationships. SBGN is composed of a set of three complementary languages: Process Description (PD), Activity Flow (AF) and Entity Relationship (ER).

Figure 8.9.: Workflow of the STON framework [Touré et al., 2016] : an SBGN-ML file is provided as the input to the framework. It is parsed by STON and converted into a graph representation using mapping rules. The resulting data is then stored in a local directory as nodes, relationships and properties. Neo4j relies on this repository and, if run as a web server instance, offers a visualization of the data. The repository can be queried for biological entities, relations in the network, and similar nodes across networks.

Figure 8.9 exemplifies the translation of a small PD map: The SBGN diagram shows a receptor complex composed of four entities: a) the IFNGR1 macromolecule, b) the IFNGR2 macromolecule, c) the JAK1 macromolecule and d) the JAK2 macromolecule. For instance, the IFNGR1 macromolecule will be translated into a node with a label macromoleculemultimer and with properties Name equals IFNGR1 and UnitOfInformation equals N:2, indicating that this node is a dimer.

STON exploits the power of graph databases to store and query complex biological pathways. This advances the possibility of:

Identifying entities in a network: When analyzing disease pathways, it is highly important to find disease-associated genes or substructures responsible to understand the organisation of a biological system and the underlying mechanism. Therefore, for a given biological entity, it is necessary to identify functionally associated network neighbourhoods, i.e. to extract a target entity and its immediate neighbours from the graph.

Linking levels of granularity: The linking of different levels of granularity allows researchers to compare biological networks at different levels of detail. In

systems biology, it is highly beneficial to have access to computational tools that can return detailed information about processes occurring in complex biological networks by connecting information from multiple layers.

Linking identical processes in two different SBGN PD diagrams: STON is capable of highlighting overlapping structures between two different metabolic graphs (PD level) by identifying and linking identical processes.

STON manages heterogeneous data at different levels of biological description by integrating i) various types of biological concepts including metabolites, proteins, complexes, genes, subcellular location, and ii) different types of processes such as metabolic reactions, signalling events and gene regulatory machinery. Once represented in a Neo4j database, the networks can be interrogated for different topics of interest using the Cypher query language [Robinson et al., 2013]. Cypher allows for structure-based queries that cannot be answered efficiently on the SBGN-ML file level, nor using SQL databases.

STON provides new opportunities for managing and querying biological networks, plus advanced manipulation of subnetworks. This will add to the infrastructure of tools for model management and exploration, which is necessary for efficient use of network approaches in systems biology and systems medicine.

8.5. CovidGraph

The COVID-19 pandemic has changed life across the globe. In January 2020, little was known about SARS-COV-2, but the vastly increasing number of infections and the uncontrolled spreading demanded fast medical action. Within a year, over 4 million publications relating to COVID-19 appeared in the scientific literature. Additionally, patents have been registered, ontologies have been extended, simulation studies for prediction of disease spread and underlying bioinformatics mechanisms have been built, and health studies have been designed. To foster integration and access to COVID-19 relevant data, Gütebier et al. [2022] developed an integrative approach to incorporate systems biology models into the domain of medical and biomedical knowledge.

CovidGraph was built as research and communication platform that encompasses publications, case statistics, genes and functions, molecular data and much more. It aims to help researchers quickly and efficiently find their way through COVID-19 datasets using tools that implement artificial intelligence methods, advanced

visualization techniques, and intuitive user interfaces. Through CovidGraph users can explore papers, patents, treatments and medications covering the family of corona viruses. In addition to literature data, information from fundamental entities in biology were connected, namely genes and proteins and their function, spanning a network of unparalleled size and knowledge (see Figure 8.10).

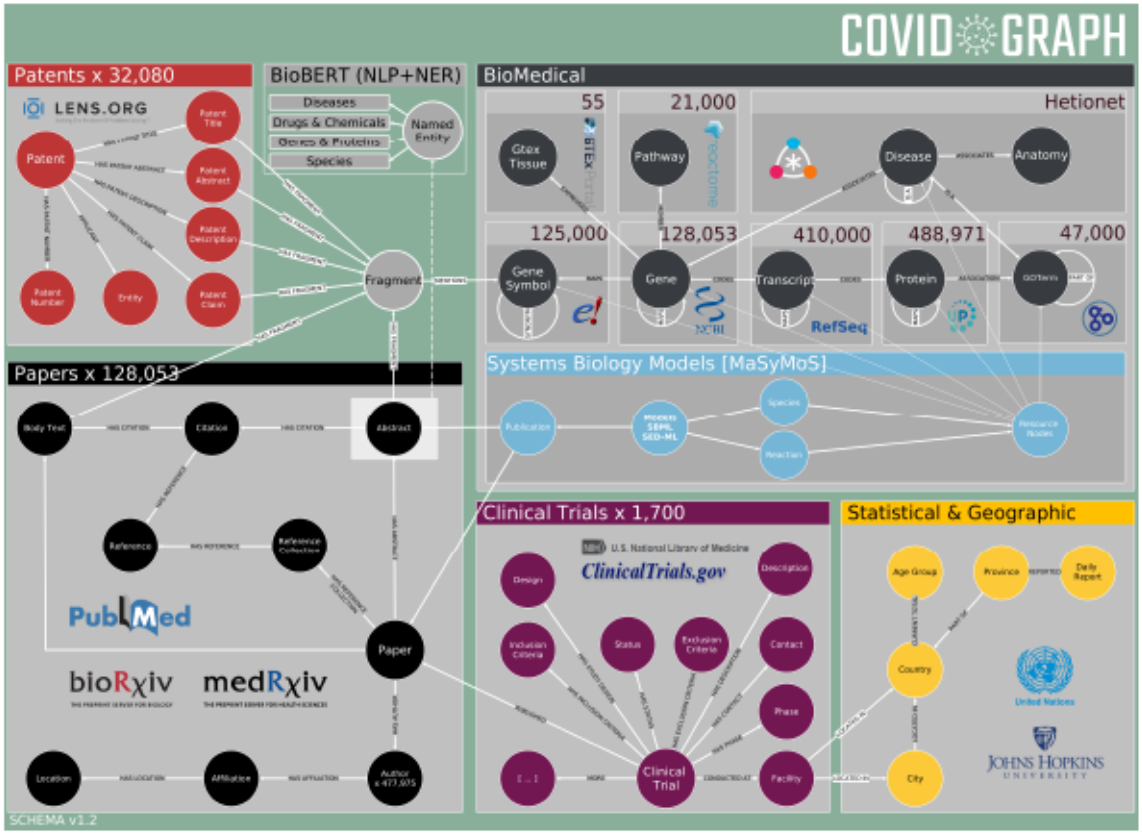


Figure 8.10.: The CovidGraph data model provides an abstract depiction of the different data sources and domains included in CovidGraph. It also shows the relations spanning different domains.

Relational databases had been traditionally developed for tabular data, which best represents homogeneous data structures. A core concept of relational databases is their fixed schema. Relational databases are not an optimal choice when it comes to storing heterogeneous, highly connected data. Over the last years, NoSQL approaches, such as Key-Value Stores, BigTable [Ghemawat et al., 2003], document databases, triple stores, or graph databases [Angles and Gutierrez, 2008], together with semantic web applications, became more popular within the life sciences [Splendiani et al., 2012]. Graph databases offer a storage concept based on nodes, (directed) edges, properties and labels. Nodes can be labeled and are connected by edges, while both can contain properties. Graph databases allow easy horizontal scaling and a fast

graph traversal. In addition, graph databases are schema optional [Lal, 2015] - a feature that is highly appreciated when storing heterogeneous, highly connected data items from different domains and sources. The CovidGraph project integrates heterogeneous resources and compiles a knowledge-base for COVID-19 and potentially other diseases. The underlying graph database of choice is Neo4j [Robinson et al., 2013], the de-facto standard.

The current version of CovidGraph[4] integrates data from five categories (Patents, Papers, BioMedical (ontologies and controlled vocabularies), Clinical Trials and Statistical & Geographic, cmp. 8.10). Categories are cross-linked by relationships. For example, papers from PubMed, medRxiv and bioRxiv are linked to the COVID-19 patent dataset from "The Lens"[5]. Major data resources include the following: The Lens provides datasets of patent documents and literature concerning human corona viruses and COVID-19. In the CovidGraph it represents the data source for patents. The COVID-19 Open Research Dataset (CORD-19) is a collection of research papers relating to COVID-19 (and corona viruses); the main data source for information about papers in the CovidGraph. It contains publications from PubMed and pre-prints from medRxiv and bioRxiv [Wang et al., 2020]. The papers and related information are stored and linked in multiple nodes in the Covid-Graph. Each paper node has author nodes connected to affiliation nodes that, in turn, are linked to location nodes. The CovidGraph contains information on clinical COVID-19 studies registered in the ClinicalTrials.gov database. It includes clinical trials nodes that are linked to multiple other nodes representing information about the clinical trial. Also, included in the CovidGraph are case statistics and case data from Johns Hopkins University [Dong et al., 2020] and population estimates and projections from the United Nations World Population Prospects[6]. Nodes in this section include city, country, province, daily report and age group. The biomedical data includes information about genes, proteins, pathways and different diseases associated with COVID-19. The data comprises information from various biological and biomedical resources and is connected to Gene Ontology terms. Information about genes from the NCBI Gene Database [Brown et al., 2015] is stored in Gene nodes which are connected to other nodes describing the underlying biology. Therefore, the connected nodes include Gene Symbols according to the Ensembl Genome Browser, a genome database [Hubbard et al., 2002]. The gene symbols are mapped to synonyms. Since

[4]CovidGraph: `https://covidgraph.org/`
[5]The Lens: `https://about.lens.org/covid-19/`
[6]United Nations World Population Prospects: `https://population.un.org/wpp/`

genes are expressed in various tissues the Gene nodes are linked to Gtex Tissue nodes containing gene expression data from the GTEx Portal. The GTEx Portal is a tissue bank and database for tissue-specific gene expression data [Lonsdale et al., 2013]. For genes that are part of a pathway there exists a relation between the corresponding gene node and pathway node. The data included in the COVID-Graph describes which genes are members of a pathway according to the Reactome pathway knowledgebase, a database for molecular information about biological pathways [Jassal et al., 2020]. As components of the transcription and translation process in humans genes code for transcripts which in turn code for proteins. These processes are described in the COVID-Graph by relationships between gene nodes, transcript nodes and protein nodes. The data for the transcript nodes is taken from the NCBI Reference Sequence Database [Pruitt et al., 2007]; the Universal Protein Resource (UniProt) provides a resource of protein sequences and annotation data [Consortium, 2019]. Proteins associated with annotation data from the Gene Ontology [Ashburner et al., 2000, Consortium, 2021] are linked to GO term nodes. The last node type that is connected with the gene nodes is information about diseases. Disease nodes are in turn associated with anatomy nodes. The corresponding data is provided by Hetionet, an integrative network of biomedical data including connections between diseases and anatomies [Himmelstein et al., 2017]. Knowledge is primarily centered around the domain of corona-viruses but is steadily extended to other connected diseases.

8.5.1. Infrastructure

The CovidGraph infrastructure is built as a labelled property graph based on the Neo4j Enterprise edition v4.2. Textual information, such as publications, clinical studies or ontology term descriptions, is enriched and recognized by a pipeline based on natural language processing and named entity recognition (BioBERT [Lee et al., 2020]). The graph, as of now, contains 15 million nodes and 32 million relationships but is still growing as the modular software framework encourages to add and integrate new data sources. Serverwise, CovidGraph relies on Docker Container. To integrate a new datasource, it needs to be wrapped in a container and provide information such as connection data and mapping information[7]. The so-called motherload[8], an ETL-process, subsequently extracts the data from the new source, transforms the data in accordance with the provided mapping information

[7]Data template: `https://github.com/covidgraph/data_template`
[8]Motherload entry point: `https://git.connect.dzd-ev.de/dzdtools/motherlode`

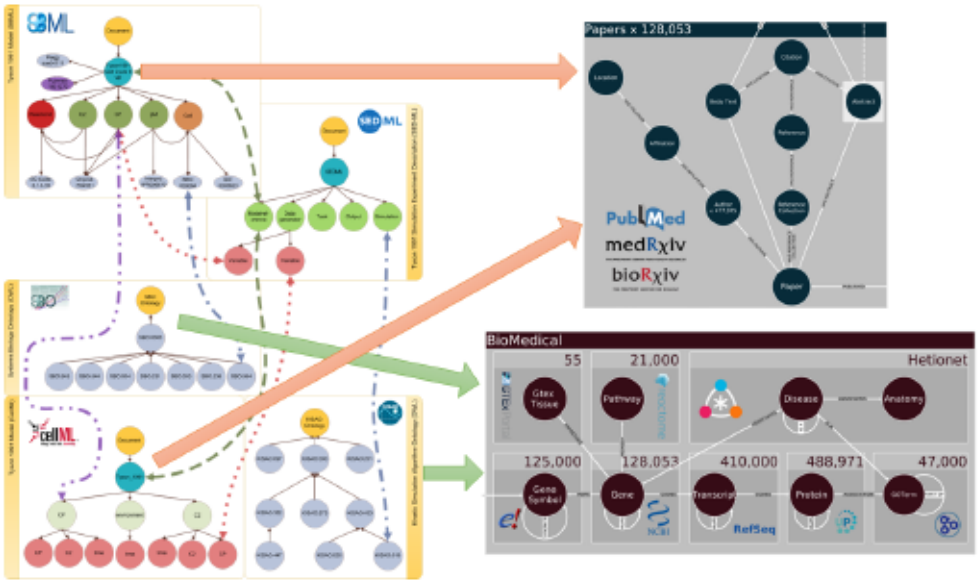


Figure 8.11.: Connections between simulation model nodes in MaSyMoS (left) and BioMedical and Papers nodes stored in CovidGraph (cmp. Fig. 8.10). Orange arrows show links between models (accompanied by a publication); green arrows denote links between ontology terms in both resources.

and loads the data into the main CovidGraph.

8.5.2. Interfaces and Availability

CovidGraph enables easy browsing across different data sources. Data exploration can be started via one of four user interfaces or programmatically via an API. Each interface is tailored towards a specific exploration approach. The interfaces as described below can be accessed through the CovidGraph website[9].

The *Visual Graph Explorer* by yWorks provides a variety of predefined views for an intuitive keyword-based graph exploration (see Figure 8.12). No prior knowledge of database query languages or the underlying graph structure is necessary for this straightforward approach. Users can interact with the clear interface by entering keywords in the search bar and filtering them for entities such as papers, patents, genes and more. Results are displayed as easily understandable connected glyphs allowing for a customised view of the selected data. When selecting a glyph (e.g, the gene ACE2), the user can load all related information (e.g., all encoded proteins). On request, additional information about the returned glyphs is provided in the detail panel.

[9]CovidGraph website: `https://healthecco.org/covidgraph/`

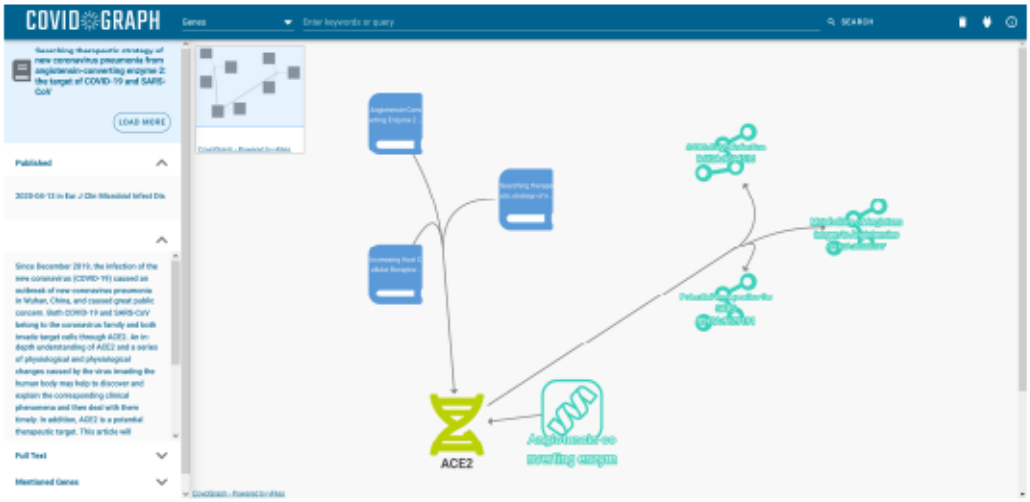


Figure 8.12.: Screenshot of the Visual Graph Explorer. The figure displays a diagram of the COVID-related gene ACE2 with encoded proteins, related publications and pathways as connected glyphs. Each entity is represented by a glyph, e.g. the entity gene is pictured as a DNA symbol. Arrows between glyphs indicate an underlying relationship between the entities. The detail panel on the left includes information about a selected publication. Keywords can be conveniently entered in the search bar on the top and filtered for various entities.

SemSpect [Liebig et al., 2017] offers unfiltered access to the database via an interactive drag-and-drop exploration tool (see Figure 8.13).

Data items selected by the user are automatically aggregated in groups (e.g., synonyms for the gene ACE2), and relations between different groups are displayed in an expandable tree structure. The user can filter, select and highlight single nodes inside a group (e.g., the gene ACE2), thereby building a tailored visual representation of the data without detailed knowledge of the underlying graph structure or query language. The resulting, customised visualisation can be exported.

Experienced users may access the database directly through Cypher queries and in combination with visual exploration, offering more sophisticated access to the data. The interface is provided by the build-in *Neo4j Browser*[10]. More precisely, the integrated command line facilitates pattern matching on the graph database by entering specific queries for patterns and paths. Query results are returned as a visual representation of the resulting graph or, depending on the user request, as a tabular format or as attribute-value pairs. The interface also includes the Graph Data Science (GDS) [11] and Awesome Procedures On Cypher (APOC) libraries [12].

[10]CovidGraph Neo4j Browser interface: `https://db.covidgraph.org/browser/`
[11]Neo4j Graph Data Science Library: `https://neo4j.com/product/graph-data-science-library/`
[12]Neo4j Awesome Procedures On Cypher: `https://neo4j.com/developer/neo4j-apoc/`

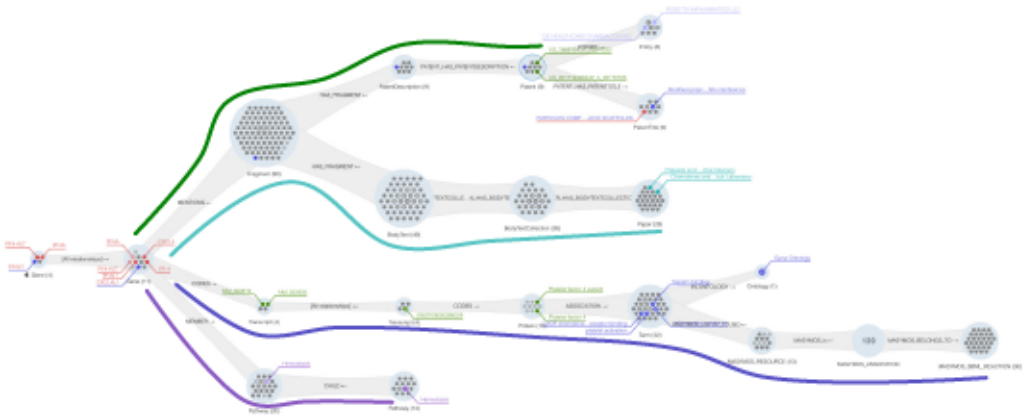


Figure 8.13.: SemSpect tree representation (enriched with colored lines). The figure displays a tree structure starting with the gene PF4 (on the left) and its synonyms. Related patents, publications, proteins, models and pathways are shown in different tree branches. Synonyms of the gene PF4 are mentioned in text fragments. These fragments are part of a patent description (green line), or belong to a text written in a paper (cyan line). The longest branch crosses multiple data domains, including biological entities, ontology terms, and simulation models (blue line). A hierarchy of pathways containing the gene PF4 or its synonyms can be seen in the lowest branch (purple line). Rather than showing relationships between nodes within a group, the interface displays relationships across different domains and aggregates them according to their type. Selected nodes are highlighted with a tag, e.g., displaying gene names.

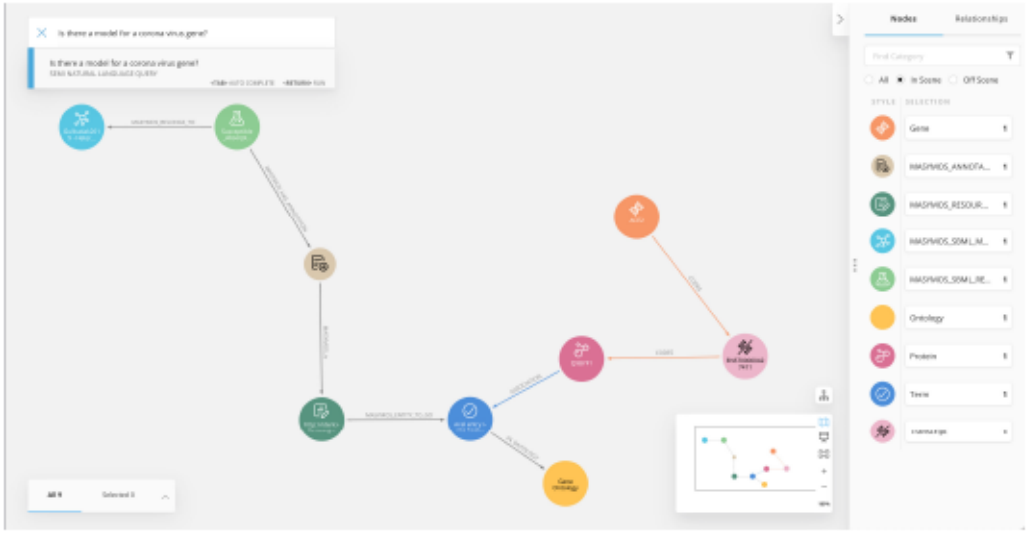


Figure 8.14.: Screenshot of Neo4j Bloom with semi-natural language query "Is there a model for a corona virus gene?". The figure shows the shortest path between the systems biology model "Gulbudak2019.1 - Heterogeneous viral strategies promote coexistence in virus-microbe systems (Lytic)" (cyan, BioModels, BIOMD0000000845) by Gulbudak and Weitz which is connected to MaSyMoS_Reaction (light green), MaSyMoS_Annotation (ochre), MaSyMoS_Resource (green) to the Gene Ontology term "Virus entry into host cell" (blue, GO:0019063) which in turn is associated with the protein Q9BYF1 (plum, UniProt) coded by the transcript ENST00000427411 (pink, RefSeq) of the corona virus gene ACE2 (orange, GenBank, ENSG00000130234).

Both allow the user to natively use common graph algorithms (e.g., community detection, link prediction, centrality, Dijkstra) or a variety of data import, export and manipulation methods, respectively. *Neo4j Bloom* is an easy-to-use graph exploration application for visually interacting with Neo4j databases (see Figure 8.14).

Neo4j Bloom provides easy-to-configure graph visualisation, as well as rule-based styling for nodes or relationships. Using semi-natural language queries allows non-computer-scientists to easily query the graph databases by typing phrases and sentences into a search bar. In addition, the Neo4j API is available for a programmatic access. It enables connection with external tools and programs for automatic querying or downloading of the data.

Use of open data and open software tools, including easy access and exploration of the integrated, heterogeneous COVID-19 data, make the CovidGraph a FAIR resource [Wilkinson et al., 2016]. The user is provided with a **F**indable, **A**ccessible, **I**nteroperable, **R**euseable, easy and intuitive data exploration tool. CovidGraph's open-source policy and free accessibility allow for an unrestricted usage and, in fact, for integration of further data sources. Its modular framework encourages the addition of new data sources and supports regular/weekly updates of the integrated data.

If it's a good idea, go ahead and do it. It's much easier to apologize than it is to get permission.

— *Grace Hopper*

9. A tale to be concluded

The story of Euclides Deanicus (see Chapter 1.1) about research lost and later retrieved also holds true for the research described in this thesis and this thesis itself, although for different reasons. This thesis began with the challenges of model retrieval [Henkel et al., 2010], successfully traversed the pitfalls of version control [Waltemath et al., 2013a] and, subsequently, even lead to the development and implementation of a model storage concept [Henkel et al., 2015] (see Figure 1.3).

Retrospectively, one could argue that, although the developed methods are scientifically sound, published in (mostly) open access peer-reviewed journals, implemented and available as open-source software and recognized by the community, the research was lost when this author left research. The reasons for this loss are of speculative nature, but two key points can be assumed: (1) the demand for the solutions proposed in this thesis were not high enough or other existing solutions covered the basic demands and (2) next to be FAIR, in order to use the developed methods and software, also a continuous development, support and advertising is necessary. Especially the continuous development and ongoing support are a fundamental problem in a project-focused, fast moving and grant dependent research environment where the main work force, the researchers, work on fixed-term contracts.

However, with the emerging threat of the Covid-19 pandemic and, simultaneously, this author being back in research, both aforementioned key points changed. In order to fight the pandemic in a timely manner, data sharing and management (including model sharing and management) became an essential task and, in addition, a hard requirement for grand applications. Subsequently, the methods, approaches and software developed in context of this thesis were remembered, retrieved and reused for Covid-19 research [Gütebier et al., 2021, Gütebier et al., 2022].

This last chapter brings this thesis, spanning 12 years of research, to an end.

Section 9.1 draws the discussion back to the research questions stated in Section 3.1 and summarizes what has been achieved. Section 9.2 elaborates on limitations of the research presented in this thesis and Section 9.3 concludes this thesis providing a perspective on future work and the way ahead.

9.1. Research questions

The research questions in this thesis have implicitly always been how to establish model management and how to make models **F**indable, **A**ccessable **I**nteroperable and **R**eusable - notwithstanding that, for the first time, an explicit definition of model management, model reuse, standards, minimum information guidelines and simulation studies was provided by Waltemath and Wolkenhauer [2016], and the FAIR principles, as guidelines for scientific data management and stewardship, were first defined by Wilkinson et al. [2016].

From the viewpoint of Waltemath and Wolkenhauer [2016] and Wilkinson et al. [2016], the aim of this thesis was to establish a baseline for a FAIR model management, taking into account model storage, ranking and retrieval and model version control (RQ1, see Table 1.1), as well as expanding the horizon and elucidate on how the aforementioned techniques of storage, retrieval, and version control can foster model reuse and allow for applications formerly not possible such es model exploration, data analysis and integration of knowledge represented by systems biology models into formerly disconnected domains (RQ6).

Model search: Finding the inhabitants. Chapter 4, based on Henkel et al. [2010], describes an application of information retrieval techniques to establish model retrieval and ranking. The driving research question (RQ2) was how heterogeneous, semi-structured model information can be retrieved and ranked.

The developed approach incorporates information encoded in the models itself and enriches these information with knowledge from external resources, such as annotations using ontologies or textual descriptions, referenced by the model. By creating and using multiple indices and a two step query expansion technique, models can be retrieved and ranked according to search terms provided by a user. Although this approach is based on adapted techniques from Information Retrieval, it was the first time that **heterogeneous, semi-structured information** about systems biology models and enriched by accompanying meta-data, could be used to easily search for, and retrieve a ranked list of matching models (RQ2). This was the first

step towards better search capabilities in model databases and expected to have a positive effect on finding and reuse of existing models directly referencing the **F**indable and **R**euseable principles of FAIR.

Model Provenance: An inhabitants life cycle. Chapter 5, based on Waltemath et al. [2013a], describes the first approach towards version control in systems biology models and explains how version control can improve model reuse. This is an essential step regarding the **R**euse principle of FAIR. The underlying research question (RQ3) was to argue why every model version should be accessible and elucidate the importance of being able to follow and understand a model's life cycle by knowing what was changed, why, and by whom.

A prerequisite to reuse a model is to first understand a model - this comprises all facets including a models history as it is common that during its life cycle, a model is subject to a characteristic set of changes, reflecting upon the model's updates from its creation to curation, publication and following updates in case of new insights or errors found. Therefore, chapter 5 advocates that all model versions should be accessible and provides reasons why changes between model versions should be presented to the user, accompanied with information what was changed, by whom and why. In addition, an algorithm to detect model changes entity-based on the level of model encoding (in this case SBML) is suggested and explained. With the idea presented here, a version control system for SBML encoded models became available for the first time. The approach developed was later refined and expanded by Scharm et al. [2016] and Scharm et al. [2018].

Model Storage: Accommodation for inhabitants. By definition a repository is a central location in which data is stored and managed. Following this idea, a repository for systems biology models can range from a pure data heap, browseable by filename, to a semantically fully integrated model storage, making a model and all accompanying data easily accessable. Obviously, the latter is to be preferred. Chapter 6, based on Henkel et al. [2015], describes the first approach towards a semantically fully integrated model storage based on graph database techniques. The driving research question (RQ4) was how to define a storage concept reflecting a model's structure and incorporate accompanying meta-data without loosing semantic information.

The described approach of MaSyMoS (Management System for Models and Simulations) is the cornerstone of this thesis. It describes a solution to the question how

a model repository can be constructed that satisfies the key points defined in Waltemath and Wolkenhauer [2016], follows the FAIR principles [Wilkinson et al., 2016], allows for easy model retrieval and ultimately fosters model reuse? In terms of the FAIR principles, this MaSyMoS fosters reusability, interoperability and accessibility by providing models, allowing to access model entities and incorporating a models semantic annotations and simulation descriptions as well as connecting different types of model-related data. By using a graph database engine, the approach is fast, flexible and can easiely be adapted to accommodate new model formats or meta-data. MaSyMoS is also the base for later model data analysis [Alm et al., 2015, Lambusch et al., 2018] and was ultimately integrated in CovidGraph [Gütebier et al., 2021, Gütebier et al., 2022] providing essential systems biology information as an addition to medical information relevant to the Covid-19 disease.

Model aspects: A model's nature. Chapter 7, based on Henkel et al. [2016a], elucidates so called model aspects (i.e., underlying encoding, references to biological entities, quantitative behavior, qualitative behavior, mathematical equations plus parameters and network structure). The driving research question (RQ5) was to elaborate what parts of a model are important and how similarity between models can be defined regarding the aforementioned aspects.

This chapter provides, on a meta-level, insight and justification about what the important bits and pieces of a model and its accompanying meta-information are and consequently allows for a decision on what and how to store (Chapter 6) and what to search for (Chapter 4) in terms of model management. It also provides information about additional methods for model comparison which are not in the focus of this thesis. Subsequently, it explains existing methods for the comparison of models, it introduces quantitative measures for model similarity and it discusses potential applications of combined similarity measures and it describes a theoretical approach to defining and computing similarities based on a combination of different model aspects.

State of the Art: Habitat status report. Chapter 8 is based on various publications and explains how the research described in this thesis has been further developed and used and what else can be achieved having a semantically integrated model storage and retrieval at hand (RQ6). The chapter elaborates on the development of model retrieval, ranking and rank aggregation on the basis of the implemented graph-based model storage, the proceedings in model version control and provenance, storage of

SBGN encoded models, and data analysis using the available model data stored in MaSyMoS.

Given the recent developments regarding the Covid-19 pandemic, Section 8.5, based on Gütebier et al. [2021], Gütebier et al. [2022] describes how MaSyMoS itself was reused and integrated as a part of CovidGraph - a much broader graph database where users can explore papers, patents, treatments and medications covering the family of corona viruses. By incorporating MaSyMoS into CovidGraph it became possible to connect systems biological models to corona-related health studies and publications and to connect model entities to genes and proteins and their functions relevant for corona research. This is an important step towards bridging the gap between systems biology and medical informatics and an excellent example of model reuse itself.

Establishing a baseline for FAIR model management. Research question 1 (RQ1) on how to establish a baseline for FAIR model management was the initial idea and driver for this thesis. However, such a broad question can not be easily answered and should be viewed more as a topic then a question.

As outlined above, considering the developed methods, approaches and prototypes to answer each defined RQ (RQ2 - RQ6) individually, already constitutes significant technological advancements and a valuable scientific knowledge gain. However, connecting the described scientific insights and advances furthermore allowed to start elaborating on the highly complex RQ1. By integrating MaSyMoS into CovidGraph the interoperability of the model storage prototype has been proven and at least one possible baseline for FAIR model management has been established.

9.2. Limitations of this research

The focus of this thesis is research and providing proof of concept for the proposed ideas and methods. Research is never all-encompassing and always holds explicit and implicit assumptions. This section elaborates the most important assumptions and their impact on the validity of this research.

Concepts and methods used, such as systems biology models, Neo4j, etc. have been chosen based on state-of-the-art availability reasons and with respect to a Pareto optimality for FAIR principles. However, methodologically, no research was done within this thesis nor is known to the author to proof that all chosen concepts are the most suitable ones, as such would have gone far beyond the scope of this thesis.

Choice of models and model encoding. This thesis deals with models, model encoding and simulation descriptions widely known and used in the Combine[1] community. Within this community, systems biology models are mostly encoded as reaction networks in SBML and CellML (and later on SBOL [Madsen et al., 2018]), while for simulation descriptions SED-ML is used. Although the community and standards are well known and established in the field of systems biology, a variety of other standards, ways of encoding and possible model representations do exist but are not considered in this thesis. Methodologically, no research was done or is known to this author to proof that the aforementioned models and model encodings are the most valuable or suitable ones for this thesis.

Model storage and retrieval concept. To map the model encoding, a graph database (Neo4j) was chosen. Although Neo4j is the de-facto standard in terms of graph databases, other graph storage concepts and implementations (including graph-relational hybrids) are available. Methodologically, no research was done or is known to this author to proof that the chosen concept and implementation of Neo4j is the most suitable one.

To enhance interoperability and reusability of MaSyMoS, the state-of-the-art standard for graph databases, Neo4j, has been chosen despite other graph storage concepts and implementations being available. In addition, the mapping of the respective encoding onto the property graph concept implemented by Neo4j is tightly fitted to the respective encoding specification. This concept was discussed within the community and with experts and found to be the most suitable. It allows users familiar with the stored models to easily traverse models and formulate queries by referring to their well known structure and representation. Still, a general problem in software development is the lack of downward compatibility in many programming languages. Here, the model retrieval concept uses, adapts and extends the index mechanisms build-in and shipped with Neo4j Version 3. The release of Neo4j Version 4 completely changed the way the database management system constructs its indexes, rendering the implementation of some parts of the model retrieval concept invalid once Version 3 will not be supported further.

Continuous specification and tool development. As in nature, the environment in systems biology modelling is constantly undergoing changes. Specifications are developed further and fitted to changing demands and needs. Those changes must

[1]Combine home page: `https://combine-org.github.io/#about`

software-wise be reflected in the corresponding parsers, i.e. for SBML, CellML and SED-ML. As all software in this field is programmed as a community effort, implementing these changes can be slow. It is also not guaranteed, that a new versions are available in the necessary language (in context of this thesis, Java). For example, SED-ML Level 1 Version 3 & 4 is not natively available in Java[2] (although there are C++ bindings). In case of SBML, the specification was split into a core specification [Keating et al., 2020] that can be extend by different specification modules - for which a parser is not necessarily available.

Even if parsers are adapted to specification changes and become available in the necessary language, the model storage, retrieval and version control concepts have to be revised and the changes implemented and reflected within the software programmed in context of this thesis - a hard to achieve task for a single researcher and programmer. In addition, the libraries, tools and APIs used by the software created in context of this thesis (most importantly Neo4j) are also constantly undergoing changes.

To ease the reuse of the "Management System for Models and Simulations", it was dockerized and documentation is provided[3]. However, it is still hard to be reused without a thorough knowledge of the models, model encoding, or the ontology mapping process. As mentioned before external APIs may be changed without notification, rendering the current docker implementation non-usable. As this thesis is written within the context of (model) reuse and reproducibility, providing hard to reuse software is contradictionary, not desirable, but in the long term inevitable.

9.3. What remains open

Research is a continuous and ongoing process, once a research question is answered and insight gained, new questions open up. This, naturally, holds true for the research done in context of this thesis.

9.3.1. Challenges and opportunities on the technological and conceptual level

Research always faces challenges and opportunities. Those challenges and opportunities can be of conceptual or technological nature and can be sketched as follows:

[2]Sed-ML tools overview: `https://sed-ml.org/showcase.html` (28.08.2022)
[3]MaSyMoS: `https://masymos.readthedocs.io/en/latest/index.html`

Incorporate new models: Next to Biomodels and PMR2, other databases and repositories containing models do exist or are being developed. BiGG [King et al., 2015], JWS [Peters et al., 2017] and Fairdom [Wolstencroft et al., 2016] seem to be suitable candidates to be included.

Incorporate new encodings and formats: While SBML, CellML and SED-ML already cover a lot of available models, new encodings such as SBOL or, more generally, disease maps [Mazein et al., 2018] would offer an extensions of knowledge.

Interfaces: Exploring and thus understanding the models and their connections as represented in the graph database can provide valuable insight for researchers. Currently, there are no sophisticated interfaces for model exploration implemented. Similar to CovidGraph, interfaces such as Neo4j Bloom or SemSpect [Liebig et al., 2017] can provide a leverage to model exploration and foster model reuse and reproducibility.

Develop analysis and prediction methods: Although some analysis methods [Alm et al., 2015, Lambusch et al., 2018] were developed, the database might offer valuable data for a deeper analysis and also predictions. Focusing on model structure and model annotation seems most promising. For example, an (at least) semi-automatic annotation prediction system helping to suggest possible annotation for non-curated models on the basis of already existing annotations and model structure will be immensely helpful.

This list is not concluding and may be extended in multiple directions depending on view point and research interests.

9.3.2. Opportunities to strengthen the research impact and practical use

To contribute to reuse and reproducibility of computational models, concepts and implementations for model storage, retrieval and version control were developed in context of this thesis. With the outbreak of Covid-19, researching SARS-CoV-2 became an immanent and important task in medical informatics and systems biology.

A lot of data, including biomedical research data, has become available since the outbreak. Collecting, storing and connecting this data spanning domains is essential in the understanding of the virus and the fight against the pandemic [Gütebier

et al., 2022]. Interestingly, the challenges arising are similar to those researched in context of this thesis. Thus, as described in Section 8.5, the storage and retrieval concept for systems biology models was adapted and integrated into the CovidGraph. As a result, the knowledge graph now contains and connects publications, patents, clinical trials, biomedical data such as genes and other molecular data, statistical and geographical information, and systems biology data in an constantly maintained and online accessible Neo4j graph database. In addition, CovidGraph already has sophisticated data exploration interfaces implemented and thus, at least partially, allows for data exploration and querying on systems biology models. As CovidGraph also offers a pipeline for easy data integration, the database can easily be extended to incorporate new model encodings, disease maps and other data or meta-data deemed to be relevant.

Integrating concepts and implementations originally developed for systems biology models into a much broader disease graph is the next step to foster model reuse and reproducibility in a scientifically wider context. As a glance into the future, this integration also offers the opportunity to tighter connect the domains of systems biology, systems medicine and medical informatics and can be seen as a first step to bridge the gap between those domains of rising importance.

Bibliography

Sibel Adali, Brandeis Hill, and Malik Magdon-Ismail. Information vs. robustness in rank aggregation: Models, algorithms and a statistical framework for evaluation. *Journal of Digital Information Management*, 5(5):292, 2007.

Richard R Adams. Sed-ed, a workflow editor for computational biology experiments written in sed-ml. *Bioinformatics*, 28(8):1180–1181, 2012.

Gerald Alexanderson. About the cover: Two theorems on geometric constructions. *Bulletin of the American Mathematical Society*, 51(3):463–467, 2014.

Rebekka Alm, Dagmar Waltemath, Olaf Wolkenauer, and Ron Henkel. Annotation-based feature extraction from sets of sbml models. In *Proceedings of the 10th international conference on Data Integration in the Life Sciences*, volume 8574, pages 81–95. Springer, 2014.

Rebekka Alm, Dagmar Waltemath, Markus Wolfien, Olaf Wolkenhauer, and Ron Henkel. Annotation-based feature extraction from sets of sbml models. *Journal of biomedical semantics*, 6(1):20, 2015.

Uri Alon. Biological networks: the tinkerer as an engineer. *Science*, 301(5641): 1866–1867, 2003.

Uri Alon. *An introduction to systems biology: design principles of biological circuits.* Chapman and Hall/CRC, 2006.

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

Rolf Apweiler, Amos Bairoch, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, et al. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl_1):D115–D119, 2004.

Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25, 2000.

Gary D. Bader and Michael P. Cary. *BioPAX - Biological Pathways Exchange Language Level 2, Version 1.0 Documentation.* BioPAX workgroup, December 2005.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1st edition, May 1999. ISBN 020139829X.

Daniel A Beard, Randall Britten, Mike T Cooling, Alan Garny, Matt DB Halstead, Peter J Hunter, James Lawson, Catherine M Lloyd, Justin Marsh, Andrew Miller, et al. Cellml metadata standards, associated tools and repositories. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1895):1845–1867, 2009.

Sean Bechhofer, Iain Buchan, David De Roure, Paolo Missier, John Ainsworth, Jiten Bhagat, Philip Couch, Don Cruickshank, Mark Delderfield, Ian Dunlop, et al. Why linked data is not enough for scientists. *Future Generation Computer Systems*, 29 (2):599–611, 2013.

C Glenn Begley and Lee M Ellis. Drug development: Raise standards for preclinical cancer research. *Nature*, 483(7391):531–533, 2012.

François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics*, 41(5):706–716, 2008.

Johannes Berg and Michael Lässig. Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Sciences*, 101(41): 14689–14694, 2004.

Frank T Bergmann and Herbert M Sauro. Comparing simulation results of sbml capable simulators. *Bioinformatics*, 24(17):1963–1965, 2008.

Frank T. Bergmann, Jonathan Cooper, David Nickerson, Nicolas Le Novère, and Dagmar Waltemath. Simulation Experiment Description Markup Language (SED-ML): Level 1 Version 2 (Specification), 2013.

Frank T Bergmann, Nicolas Rodriguez, Nicolas Le Novère, and Babraham Campus Cambridge. COMBINE Archive Specification: Version 1 Release Candidate 2, July 2014.

Frank T Bergmann, Stefan Hoops, Brian Klahn, Ursula Kummer, Pedro Mendes, Jürgen Pahle, and Sven Sahle. Copasi and its applications in biotechnology. *Journal of biotechnology*, 261:215–220, 2017a.

Frank T Bergmann, David Nickerson, Dagmar Waltemath, and Martin Scharm. Sed-ml web tools: generate, modify and export standard-compliant simulation studies. *Bioinformatics*, 33(8):1253–1254, 2017b.

Frank T Bergmann, Jonathan Cooper, Matthias König, Ion Moraru, David Nickerson, Nicolas Le Novère, Brett G Olivier, Sven Sahle, Lucian Smith, and Dagmar Waltemath. Simulation experiment description markup language (sed-ml) level 1 version 3 (l1v3). *Journal of integrative bioinformatics*, 15(1), 2018.

Anna Bernasconi and Marco Masseroli. Biological and medical ontologies: Systems biology ontology (sbo). In Shoba Ranganathan, Michael Gribskov, Kenta Nakai, and Christian Schönbach, editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 858 – 866. Academic Press, Oxford, 2019. ISBN 978-0-12-811432-2. doi: https://doi.org/10.1016/B978-0-12-809633-8.20399-3.

T.D. Brock, M.T. Madigan, J.M. Martinko, and J. Parker. *Biology of microorganisms*. Prentice-Hall Englewood Cliffs, NJ, 1974.

Garth R Brown, Vichet Hem, Kenneth S Katz, Michael Ovetsky, Craig Wallin, Olga Ermolaeva, Igor Tolstoy, Tatiana Tatusova, Kim D Pruitt, Donna R Maglott, et al. Gene: a gene-centered information resource at ncbi. *Nucleic acids research*, 43 (D1):D36–D42, 2015. doi: https://doi.org/10.1093/nar/gku1055.

Finja Büchel, Nicolas Rodriguez, Neil Swainston, Clemens Wrzodek, Tobias Czauderna, Roland Keller, Florian Mittag, Michael Schubert, Mihai Glont, Martin Golebiewski, et al. Path2models: large-scale generation of computational models from biochemical pathway maps. *BMC systems biology*, 7(1):116, 2013.

Peter Buneman. Semistructured data. *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 117–121, 1997.

Robert C Cannon, Padraig Gleeson, Sharon Crook, Gautham Ganapathy, Boris Marin, Eugenio Piasini, and R Angus Silver. Lems: a language for expressing complex biological models in concise and hierarchical form and its use in underpinning neuroml 2. *Frontiers in neuroinformatics*, 8:79, 2014.

S.S. Chawathe et al. Change detection in hierarchically structured information. In *ACM SIGMOD Record*, volume 25, pages 493–504. ACM, 1996.

Vijayalakshmi Chelliah, Lukas Endler, Nick Juty, Camille Laibe, Chen Li, Nicolas Rodriguez, and Nicolas Le Novère. Data integration and semantic enrichment of systems biology models and simulations. In *Proceedings of the 6th international conference on Data Integration in the Life Sciences*, volume 5647, pages 5–15. Springer, 2009.

Vijayalakshmi Chelliah, Camille Laibe, and Nicolas Le Novère. Biomodels database: A repository of mathematical models of biological processes. In *In Silico Systems Biology*, pages 189–199. Springer, 2013.

Vijayalakshmi Chelliah, Nick Juty, Ishan Ajmera, Raza Ali, Marine Dumousseau, Mihai Glont, Michael Hucka, Gaël Jalowicki, Sarah Keating, Vincent Knight-Schrijver, et al. Biomodels: ten-year anniversary. *Nucleic acids research*, 43(D1): D542–D548, 2015.

Connor Clark and Jugal Kalita. A comparison of algorithms for the pairwise alignment of biological networks. *Bioinformatics*, 30(16):2351–2359, 2014.

Gregory Cobena, Serge Abiteboul, and Amelie Marian. Detecting changes in xml documents. In *Proceedings 18th International Conference on Data Engineering*, pages 41–52. IEEE, 2002.

Gene Ontology Consortium. The gene ontology resource: 20 years and still going strong. *Nucleic acids research*, 47(D1):D330–D338, 2018.

The Gene Ontology Consortium. The gene ontology resource: enriching a gold mine. *Nucleic Acids Research*, 49(D1):D325–D334, 2021. doi: https://doi.org/10.1093/nar/gkaa1113.

UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47(D1):D506–D515, 2019. doi: https://doi.org/10.1093/nar/gky1049.

UniProt Consortium et al. The universal protein resource (uniprot). *Nucleic acids research*, 36(suppl 1):D190–D195, 2008.

UniProt Consortium et al. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 46(5):2699, 2018.

Jonathan Cooper, Gary R Mirams, and Steven A Niederer. High-throughput functional curation of cellular electrophysiology models. *Progress in biophysics and molecular biology*, 107(1):11–20, 2011.

Jonathan Cooper, Jon Olav Vik, and Dagmar Waltemath. A call for virtual experiments: accelerating the scientific process. *Progress in biophysics and molecular biology*, S0079-6107(14)00182-5, November 2014. doi: http://dx.doi.org/10.1016/j.pbiomolbio.2014.10.001.

Jonathan Cooper, Martin Scharm, and Gary R Mirams. The cardiac electrophysiology web lab. *Biophysical journal*, 110(2):292–300, 2016.

Mélanie Courtot, Nick Juty, Christian Knüpfer, Dagmar Waltemath, Anna Zhukova, Andreas Dräger, Michel Dumontier, Andrew Finney, Martin Golebiewski, Janna Hastings, et al. Controlled vocabularies and semantics in systems biology. *Molecular systems biology*, 7(1):543, 2011.

A.A. Cuellar, C.M. Lloyd, P.M.F. Nielsen, D.P. Bullivant, D.P. Nickerson, and P.J. Hunter. An overview of cellml 1.1, a biological model description language. *Simulation*, 79(12):740–747, 2003.

Automn A Cuellar, Melanie Nelson, Warren Hedley, D Bullivant, F Livingston, C Lloyd, and P Nielsen. The cellml metadata 1.0 specification, 2002.

Tobias Czauderna, Christian Klukas, and Falk Schreiber. Editing, validating and translating of sbgn maps. *Bioinformatics*, 26(18):2340–2341, 2010.

Joseph O Dada, Irena Spasić, Norman W Paton, and Pedro Mendes. SBRML: a markup language for associating systems biology data with models. *Bioinformatics*, 26(7):932–938, 2010.

Bernard de Bono, Robert Hoehndorf, Sarala Wimalaratne, George Gkoutos, and Pierre Grenon. The ricordo approach to semantic interoperability for biomedical data and models: strategy, standards and solutions. *BMC Research Notes*, 4(1): 313, 2011.

Kirill Degtyarenko, Paula de Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, 36(suppl 1):D344–D350, 2008.

Emek Demir, Michael P Cary, Suzanne Paley, Ken Fukuda, Christian Lemer, Imre Vastrik, Guanming Wu, Peter D'Eustachio, Carl Schaefer, Joanne Luciano, et al. The biopax community standard for pathway data sharing. *Nature biotechnology*, 28(9):935–942, 2010.

Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track covid-19 in real time. *The Lancet infectious diseases*, 20(5):533–534, 2020. doi: https://doi.org/10.1016/S1473-3099(20)30120-1.

Andreas Dräger, Nicolas Rodriguez, Marine Dumousseau, Alexander Dörr, Clemens Wrzodek, Nicolas Le Novère, Andreas Zell, and Michael Hucka. JSBML: a flexible java library for working with SBML. *Bioinformatics*, 27(15):2167–2168, 2011.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern Classification*. Wiley-Interscience, 2000.

Michel Dumontier, Leonid L. Chepelev, and Robert Hoehndorf. Semantic systems biology: Formal knowledge representation in systems biology for model construction, retrieval, validation and discovery. In Aleš Prokop and Béla Csukás, editors, *Systems Biology*, pages 355–373. Springer Netherlands, 2013. ISBN 978-94-007-6802-4. doi: 10.1007/978-94-007-6803-1_12.

Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001.

Lukas Endler, Nicolas Rodriguez, Nick Juty, Vijayalakshmi Chelliah, Camille Laibe, Chen Li, and Nicolas Le Novère. Designing and encoding models for synthetic biology. *Journal of The Royal Society Interface*, 6(Suppl 4):S405–S417, August 2009. ISSN 1742-5662. doi: 10.1098/rsif.2009.0035.focus.

Maria Esch, Jinbo Chen, Christian Colmsee, Matthias Klapperstück, Eva Grafahrend-Belau, Uwe Scholz, and Matthias Lange. Lailaps: the plant science search engine. *Plant and Cell Physiology*, 56(1):e8–e8, 2014.

Scott Federhen. The ncbi taxonomy database. *Nucleic acids research*, 40(D1): D136–D143, 2011.

R. Ferber. *Information Retrieval: Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web.* dpunkt Verlag, 2003.

Ray W Fergerson, Paul R Alexander, Michael Dorf, Rafael S Gonçalves, Manuel Salvadores, Alex Skrenchuk, Jennifer Vendetti, and Mark A Musen. Ncbo bioportal version 4. In *ICBO*, 2015.

James E Ferrell. Q&A: Systems Biology. *Journal of biology*, 8(1):2, 2009.

Anthony Finkelstein et al. Computational challenges of systems biology. *IEEE Computer*, 37(5):26–33, 2004.

Andrew Finney, Michael Hucka, and Nicolas Le Novère. Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions, 2003.

Akira Funahashi, Yukiko Matsuoka, Akiya Jouraku, Mineo Morohashi, Norihiro Kikuchi, and Hiroaki Kitano. Celldesigner 3.5: a versatile modeling tool for biochemical networks. *Proceedings of the IEEE*, 96(8):1254–1265, 2008.

Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010.

Alan Garny and Peter J Hunter. Opencor: a modular and interoperable approach to computational biology. *Frontiers in physiology*, 6:26, 2015.

Steven Gay, Sylvain Soliman, and François Fages. A graphical method for reducing and relating models in systems biology. *Bioinformatics*, 26(18):i575–i581, 2010.

John H Gennari, Maxwell L Neal, Michal Galdzicki, and Daniel L Cook. Multiple ontologies in action: Composite annotations for biosimulation models. *Journal of biomedical informatics*, 44(1):146–154, 2011.

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43, October 2003. ISSN 0163-5980. doi: 10.1145/1165389.945450.

Padraig Gleeson, Sharon Crook, Robert C Cannon, Michael L Hines, Guy O Billings, Matteo Farinella, Thomas M Morse, Andrew P Davison, Subhasis Ray, Upinder S Bhalla, et al. NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail. *PLoS computational biology*, 6(6):e1000815, 2010.

Padraig Gleeson, Matteo Cantarelli, Boris Marin, Adrian Quintana, Matt Earnshaw, Sadra Sadeh, Eugenio Piasini, Justas Birgiolas, Robert C Cannon, N Alex Cayco-Gajic, et al. Open source brain: A collaborative resource for visualizing, analyzing, simulating, and developing standardized models of neurons and circuits. *Neuron*, 2019.

Mihai Glont, Tung V N Nguyen, Martin Graesslin, Robert Hälke, Raza Ali, Jochen Schramm, Sarala M Wimalaratne, Varun B Kothamachu, Nicolas Rodriguez, Maciej J Swat, et al. Biomodels: expanding horizons to include more modelling approaches and formats. *Nucleic acids research*, 46(D1):D1248–D1253, 2017.

Albert Goldbeter. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *Proceedings of the National Academy of Sciences*, 88(20): 9107–9111, 1991.

Harold F Gómez, Michael Hucka, Sarah M Keating, German Nudelman, Dagmar Iber, and Stuart C Sealfon. Moccasin: converting matlab ode models to sbml. *Bioinformatics*, 32(12):1905–1906, 2016.

O. Gospodnetic and E. Hatcher. Lucene in action: a guide to the Java search engine. *Greenwich (USA): Manning*, 2005.

Thomas R Gruber et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

Xiang Guo, Rongxiang Liu, Craig D Shriver, Hai Hu, and Michael N Liebman. Assessing semantic similarity measures for the characterization of human regulatory pathways. *Bioinformatics*, 22(8):967–973, 2006.

Lea Gütebier, Ron Henkel, Alexander Jarasch, Tim Bleimehl, Sebastian Müller, Jamie Munro, M Preusse, Dagmar Walthemath, and Team HealthEcco. Covidgraph: Connecting biomedical covid-19 resources and computational biology models. In *2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores, SEA-Data 2021*, pages 34–37, 2021.

Lea Gütebier, Tim Bleimehl, Ron Henkel, Jamie Munro, Sebastian Müller, Axel Morgner, Jakob Laenge, Anke Pachauer, Alexander Erdl, Jens Weimar, Kirsten Walther Langendorf, Vincent Vialard, Thorsten Liebig, Martin Preusse, Dagmar Waltemath, and Alexander Jarasch. CovidGraph: a graph to fight COVID-19. *Bioinformatics*, 38(20):4843–4845, 08 2022. ISSN 1367-4803. doi: 10.1093/bioinformatics/btac592.

Janna Hastings, Gareth Owen, Adriano Dekker, Marcus Ennis, Namrata Kale, Venkatesh Muthukrishnan, Steve Turner, Neil Swainston, Pedro Mendes, and Christoph Steinbeck. Chebi in 2016: Improved services and an expanding collection of metabolites. *Nucleic acids research*, 44(D1):D1214–D1219, 2015.

Ron Henkel. Determining model similarities through ranking functions using the example of biological computational models, Sept. 2009. Diplomarbeit, Chair of Database Systems, Department of Computer Science.

Ron Henkel. Biomedical repositories for simulation studies. In *Reference Module in Biomedical Sciences*. Elsevier, 2020. doi: 10.1016/b978-0-12-801238-3.11684-8.

Ron Henkel, Dagmar Köhn, André Peters, and Carsten Maus. Similarity measures for computational biological models. In *International Workshop on Data Integration in the Life Sciences*, 2009.

Ron Henkel, Lukas Endler, Andre Peters, Nicolas Le Novère, and Dagmar Waltemath. Ranked retrieval of computational biology models. *BMC bioinformatics*, 11(1): 423, 2010.

Ron Henkel, Dagmar Waltemath, and Olaf Wolkenhauer. Managing bio-models: Model storage, retrieval, ranking and versioning, 2011.

Ron Henkel, Nicolas Le Novère, Olaf Wolkenhauer, and Dagmar Waltemath. Considerations of graph-based concepts to manage of computational biology models and associated simulations. *Proceedings of the 2012 GI-Jahrestagung*, pages 1545–1551, 2012a.

Ron Henkel, Nicolas Le Novère, Olaf Wolkenhauer, and Dagmar Waltemath. Considerations of graph-based concepts to manage of computational biology models and associated simulations. *INFORMATIK 2012*, 2012b.

Ron Henkel, Olaf Wolkenhauer, and Dagmar Waltemath. Combining computational models, semantic annotations and simulation experiments in a graph database. *Database*, 2015:bau130, 2015.

Ron Henkel, Robert Hoehndorf, Tim Kacprowski, Christian Knüpfer, Wolfram Liebermeister, and Dagmar Waltemath. Notions of similarity for systems biology models. *Briefings in Bioinformatics*, page bbw090, 2016a.

Ron Henkel, Fabienne Lambusch, Olaf Wolkenhauer, Kurt Sandkuhl, Christian Rosenke, and Dagmar Waltemath. Finding patterns in biochemical reaction networks. *PeerJ PrePrints*, 4:e1479v2, 2016b. doi: https://doi.org/10.7287/peerj.preprints.1479v2.

Markus J Herrgård, Neil Swainston, Paul Dobson, Warwick B Dunn, K Yalçin Arga, Mikko Arvas, Nils Blüthgen, Simon Borger, Roeland Costenoble, Matthias

Heinemann, et al. A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology. *Nature biotechnology*, 26(10): 1155–1160, 2008.

Kristina M Hettne, Harish Dharuri, Jun Zhao, Katherine Wolstencroft, Khalid Belhajjame, Stian Soiland-Reyes, Eleni Mina, Mark Thompson, Don Cruickshank, Lourdes Verdes-Montenegro, et al. Structuring research methods and data with the research object model: genomics workflows as a case study. *Journal of Biomedical Semantcs*, 5(41), 2014.

Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife*, 6, September 2017. doi: 10.7554/elife.26726.

Michael Hines. Neuron—a program for simulation of nerve equations. In *Neural systems: Analysis and modeling*, pages 127–136. Springer, 1993.

Michael L Hines, Thomas Morse, Michele Migliore, Nicholas T Carnevale, and Gordon M Shepherd. Modeldb: a database to support computational neuroscience. *Journal of computational neuroscience*, 17(1):7–11, 2004.

D.S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, 24(4):664–675, 1977.

Robert Hoehndorf, Michel Dumontier, John H Gennari, Sarala Wimalaratne, Bernard de Bono, Daniel L Cook, and Georgios V Gkoutos. Integrating systems biology models and biomedical ontologies. *BMC systems biology*, 5(1):124, 2011.

Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. Copasi—a complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006.

Tim Hubbard, Daniel Barker, Ewan Birney, Graham Cameron, Yuan Chen, L Clark, Tony Cox, J Cuff, Val Curwen, Thomas Down, et al. The ensembl genome database project. *Nucleic acids research*, 30(1):38–41, 2002. doi: https://doi.org/10.1093/nar/30.1.38.

Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19 (4):524–531, 2003.

Michael Hucka, Frank T Bergmann, Sarah M Keating, James C Schaff, and Lucian P Smith. The systems biology markup language (SBML): Language specification for level 3 version 1 core, 2010.

Michael Hucka, Frank T Bergmann, Sarah M Keating, and Lucian P Smith. A profile of today's sbml-compatible software. In *Proceedings of the 2011 IEEE Seventh International Conference on e-Science (eScienceW)*, pages 143–150. IEEE, 2011.

Michael Hucka, David P Nickerson, Gary D Bader, Frank T Bergmann, Jonathan Cooper, Emek Demir, Alan Garny, Martin Golebiewski, Chris J Myers, Falk Schreiber, et al. Promoting coordinated development of community-based information standards for modeling in biology: the combine initiative. *Frontiers in bioengineering and biotechnology*, 3:19, 2015.

Michael Hucka, Frank T Bergmann, Andreas Dräger, Stefan Hoops, Sarah M Keating, Nicolas Le Novère, Chris J Myers, Brett G Olivier, Sven Sahle, James C Schaff, et al. The systems biology markup language (sbml): language specification for level 3 version 2 core. *Journal of integrative bioinformatics*, 15(1), 2018.

Michael Hucka, Frank T Bergmann, Claudine Chaouiya, Andreas Dräger, Stefan Hoops, Sarah M Keating, Matthias König, Nicolas Le Novère, Chris J Myers, Brett G Olivier, et al. The systems biology markup language (sbml): Language specification for level 3 version 2 core release 2. *Journal of Integrative Bioinformatics*, 2019.

Bijay Jassal, Lisa Matthews, Guilherme Viteri, Chuqiao Gong, Pascual Lorente, Antonio Fabregat, Konstantinos Sidiropoulos, Justin Cook, Marc Gillespie, Robin Haw, et al. The reactome pathway knowledgebase. *Nucleic acids research*, 48(D1): D498–D503, 2020. doi: https://doi.org/10.1093/nar/gkz1031.

Simon Jupp, James Malone, Jerven Bolleman, Marco Brandizi, Mark Davies, Leyla Garcia, Anna Gaulton, Sebastien Gehant, Camille Laibe, Nicole Redaschi, et al. The ebi rdf platform: linked open data for the life sciences. *Bioinformatics*, 30(9): 1338–1339, 2014.

N Juty, Raza Ali, M Glont, S Keating, Ns Rodriguez, MJ Swat, SM Wimalaratne, H Hermjakob, N Le Novère, C Laibe, et al. Biomodels: content, features, functionality, and use. *CPT: pharmacometrics & systems pharmacology*, 4(2):55–68, 2015.

Nick Juty, Nicolas Le Novère, and Camille Laibe. Identifiers. org and MIRIAM Registry: community resources to provide persistent identification. *Nucleic acids research*, 40(D1):D580–D586, 2012.

Minoru Kanehisa, Susumu Goto, Yoko Sato, Miho Furumichi, and Mao Tanabe. Kegg for integration and interpretation of large-scale molecular data sets. *Nucleic acids research*, page gkr988, 2011.

Minoru Kanehisa, Miho Furumichi, Mao Tanabe, Yoko Sato, and Kanae Morishima. Kegg: new perspectives on genomes, pathways, diseases and drugs. *Nucleic acids research*, 45(D1):D353–D361, 2016.

Jonathan R Karr, Jayodita C Sanghvi, Derek N Macklin, Miriam V Gutschow, Jared M Jacobs, Benjamin Bolival Jr, Nacyra Assad-Garcia, John I Glass, and Markus W Covert. A whole-cell computational model predicts phenotype from genotype. *Cell*, 150(2):389–401, 2012.

Sarah M Keating, Dagmar Waltemath, Matthias König, Fengkai Zhang, Andreas Dräger, Claudine Chaouiya, Frank T Bergmann, Andrew Finney, Colin S Gillespie, Tomáš Helikar, et al. Sbml level 3: an extensible format for the exchange and reuse of biological models. *Molecular systems biology*, 16(8):e9110, 2020.

Zachary A King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A Lerman, Ali Ebrahim, Bernhard O Palsson, and Nathan E Lewis. Bigg models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic acids research*, 44(D1):D515–D522, 2015.

E. Klipp, W. Liebermeister, C. Wierling, A. Kowald, H. Lehrach, and R. Herwig. *Systems biology: a textbook*. Wiley-VCH, 2009.

Christian Knüpfer, Clemens Beckstein, and Peter Dittrich. Towards a semantic description of bio-models: Meaning facets - a case study. In *Proceedings of the Second International Symposium on Semantic Mining in Biomedicine*, pages 97–100, June 2006.

Christian Knüpfer, Clemens Beckstein, Peter Dittrich, and Nicolas Le Novère. Structure, function, and behaviour of computational models in systems biology. *BMC systems biology*, 7(1):43, 2013.

Dagmar Köhn and Lena Strömbäck. A method for semi-automatic standard integration in systems biology. In *Proceedings of the 19th International Conference on Database and Expert Systems Applications*, volume 5181, pages 745–752. Springer, 2008.

Dagmar Köhn, Carsten Maus, Ron Henkel, and Martin Kolbe. Towards enhanced retrieval of biological models through annotation-based ranking. In *International Workshop on Data Integration in the Life Sciences*, pages 204–219. Springer, 2009.

F. Krause et al. Annotation and merging of sbml models with semanticsbml. *Bioinformatics*, 26(3):421, 2010.

Falko Krause, Jannis Uhlendorf, Timo Lubitz, Marvin Schulz, Edda Klipp, and Wolfram Liebermeister. Annotation and merging of sbml models with semanticsbml. *Bioinformatics*, 26(3):421–422, 2009.

Falko Krause, Marvin Schulz, Neil Swainston, and Wolfram Liebermeister. Sustainable model building: the role of standards and biological semantics. In *Methods in enzymology*, volume 500, pages 371–395. Elsevier, 2011.

C. Laibe and N. Le Novère. MIRIAM Resources: tools to generate and resolve robust cross-references in Systems Biology. *BMC Systems Biology*, 1(1):58, 2007.

Mahesh Lal. *Neo4j graph data modeling*. Packt Publishing Ltd, 2015.

Fabienne Lambusch, Dagmar Waltemath, Olaf Wolkenhauer, Kurt Sandkuhl, Christian Rosenke, and Ron Henkel. Identifying frequent patterns in biochemical reaction networks: a workflow. *Database*, 2018, 2018.

Matthias Lange, Karl Spies, Joachim Bargsten, Gregor Haberhauer, Matthias Klapperstück, Michael Leps, Christian Weinel, Röbbe Wünschiers, Mandy Weissbach, Jens Stein, et al. The lailaps search engine: relevance ranking in life science databases. *Journal of Integrative Bioinformatics*, 7(2):1–11, 2010.

Matthias Lange, Ron Henkel, Wolfgang Müller, Dagmar Waltemath, and Stephan Weise. Information retrieval in life sciences: a programmatic survey. In *Approaches in Integrative Bioinformatics*, pages 73–109. Springer Berlin Heidelberg, 2014.

O. Lassila et al. Resource description framework (RDF) model and syntax specification, 1998a.

Ora Lassila, Ralph R Swick, et al. Resource description framework (rdf) model and syntax specification, 1998b.

Nicolas Le Novère. Model storage, exchange and integration. *BMC neuroscience*, 7 (1):S11, 2006.

Nicolas Le Novère. Quantitative and logic modelling of gene and molecular networks. *Nature Reviews. Genetics*, 16(3):146, 2015.

Nicolas Le Novère, Andrew Finney, Michael Hucka, Upinder S Bhalla, Fabien Campagne, Julio Collado-Vides, Edmund J Crampin, Matt Halstead, Edda Klipp, Pedro Mendes, et al. Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature biotechnology*, 23(12):1509–1515, 2005.

Nicolas Le Novère, Benjamin Bornstein, Alexander Broicher, Melanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, et al. Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic acids research*, 34(suppl 1):D689–D691, 2006.

Nicolas Le Novère, Melanie Courtot, and Camille Laibe. Adding semantics in kinetics models of biochemical pathways. In *Proceedings of the 2nd International Symposium on experimental standard conditions of enzyme characterizations*, pages 137–153. Citeseer, 2007.

Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I Aladjem, Sarala M Wimalaratne, et al. The systems biology graphical notation. *Nature biotechnology*, 27(8):735–741, 2009.

Nicolas Le Novère, Michael Hucka, Nadia Anwar, Gary D Bader, Emek Demir, Stuart Moodie, and Anatoly Sorokin. Meeting report from the first meetings of the Computational Modeling in Biology Network (COMBINE). *Standards in genomic sciences*, 5(2):230, 2011.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Chen Li, Mélanie Courtot, Nicolas Le Novère, and Camille Laibe. Biomodels. net web services, a free and integrated toolkit for computational modelling software. *Briefings in bioinformatics*, page bbp056, 2009.

Chen Li, Marco Donizelli, Nicolas Rodriguez, Harish Dharuri, Lukas Endler, Vijayalakshmi Chelliah, Lu Li, Enuo He, Arnaud Henry, Melanie I Stefan, et al. Biomodels database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC systems biology*, 4(1):92, 2010.

Yuhua Li, Zuhair A Bandar, and David McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on knowledge and data engineering*, 15(4):871–882, 2003.

W. Liebermeister. Validity and combination of biochemical models. In *Proceedings of 3rd International ESCEC Workshop on Experimental Standard Conditions on Enzyme Characterizations*, 2008.

Thorsten Liebig, Vincent Vialard, and Michael Opitz. Connecting the Dots in Million-Nodes Knowledge Graphs with SemSpect. In *International Semantic Web Conference (Posters, Demos & Industry Tracks)*, 2017.

Allyson L Lister, Phillip Lord, Matthew Pocock, and Anil Wipat. Annotation of sbml models through rule-based semantic integration. *J Biomed Semantics*, 1 (Suppl 1):S3, 2010.

Catherine M Lloyd, Matt DB Halstead, and Poul F Nielsen. Cellml: its future, present and past. *Progress in biophysics and molecular biology*, 85(2):433–450, 2004.

Catherine M Lloyd, James R Lawson, Peter J Hunter, and Poul F Nielsen. The cellml model repository. *Bioinformatics*, 24(18):2122–2123, 2008.

John Lonsdale, Jeffrey Thomas, Mike Salvatore, Rebecca Phillips, Edmund Lo, Saboor Shad, Richard Hasz, Gary Walters, Fernando Garcia, Nancy Young, et al. The genotype-tissue expression (gtex) project. *Nature genetics*, 45(6):580–585, 2013. doi: https://doi.org/10.1038/ng.2653.

Curtis Madsen, Angel Goni Moreno, P Umesh, Zachary Palchick, Nicholas Roehner, Christian Atallah, Bryan Bartley, Kiri Choi, Robert Sidney Cox, Thomas Gorochowski, et al. Synthetic biology open language (sbol) version 2.3. *Journal of integrative bioinformatics*, 2018.

Lorenzo Mascheroni. *Geometria del Compasso*. Pavia, Galeazzi, 1797.

Alexander Mazein, Marek Ostaszewski, Inna Kuperstein, Steven Watterson, Nicolas Le Novère, Diane Lefaudeux, Bertrand De Meulder, Johann Pellet, Irina Balaur, Mansoor Saqi, et al. Systems medicine disease maps: community-driven comprehensive representation of disease mechanisms. *NPJ systems biology and applications*, 4(1):1–10, 2018.

Robert A McDougal, Anna S Bulanova, and William W Lytton. Reproducibility in computational neuroscience models and simulations. *IEEE Transactions on Biomedical Engineering*, 63(10):2021–2035, 2016.

Robert A McDougal, Thomas M Morse, Ted Carnevale, Luis Marenco, Rixin Wang, Michele Migliore, Perry L Miller, Gordon M Shepherd, and Michael L Hines. Twenty years of modeldb and beyond: building essential modeling tools for the future of neuroscience. *Journal of computational neuroscience*, 42(1):1–10, 2017.

Andrew K Miller, Tommy Yu, Randall Britten, Mike T Cooling, James Lawson, Dougal Cowan, Alan Garny, Matt DB Halstead, Peter J Hunter, David P Nickerson, et al. Revision history aware repositories of computational models of biological systems. *BMC bioinformatics*, 12(1):22, 2011.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Georg Mohr. *Euclides Danicus*. Bestaende in twee Deelen, Copenhagen and Amsterdam, 1672.

Yuda Munarko, Dewan Mahabub Sarwar, Koray Atalag, and David Phillip Nickerson. Nlimed: Natural language interface for model entity discovery in biosimulation model repositories. *bioRxiv*, page 756304, 2019.

Chris J Myers, Gary Bader, Padraig Gleeson, Martin Golebiewski, Michael Hucka, Nicolas Le Novère, David P Nickerson, Falk Schreiber, and Dagmar Waltemath. A brief history of combine. In *Proceedings of the 2017 Winter Simulation Conference*, page 63. IEEE Press, 2017.

Mariam Nassar, Dagmar Waltemath, and Ron Henkel. Rank aggregation in masymos. unpublished, 2015. Technical documentation and implementation description.

Bela Novak and John J Tyson. Modeling the control of dna replication in fission yeast. *Proceedings of the National Academy of Sciences*, 94(17):9147–9152, 1997.

Natalya F Noy, Nigam H Shah, Patricia L Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L Rubin, Margaret-Anne Storey, Christopher G Chute, et al. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, 37(suppl 2):W170–W173, 2009.

Hiroyuki Ogata, Susumu Goto, Kazushige Sato, Wataru Fujibuchi, Hidemasa Bono, and Minoru Kanehisa. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 27(1):29–34, 1999.

Brett G Olivier and Jacky L Snoep. Web-based kinetic modelling using jws online. *Bioinformatics*, 20(13):2143–2144, 2004.

Stephan Pabinger, Rene Snajder, Timo Hardiman, Michaela Willi, Andreas Dander, and Zlatko Trajanoski. Memosys 2.0: an update of the bioinformatics database for genome-scale models and genomic data. *Database*, 2014, 2014.

Young M Park, Silvano Squizzato, Nicola Buso, Tamer Gur, and Rodrigo Lopez. The ebi search engine: Ebi search as a service—making biological data accessible for all. *Nucleic acids research*, 45(W1):W545–W549, 2017.

Catia Pesquita, Daniel Faria, Andre O Falcao, Phillip Lord, and Francisco M Couto. Semantic similarity in biomedical ontologies. *PLoS computational biology*, 5(7): e1000443, 2009.

Martin Peters. Storage solution for models with multiple versions in systems biology: Concept and prototype implementation. Master's thesis, University of Rostock, 18051, Rostock, 11 2016. BA Thesis, Chair of Systems Biology, Department of Computer Science.

Martin Peters, Johann J Eicher, David D van Niekerk, Dagmar Waltemath, and Jacky L Snoep. The jws online simulation database. *Bioinformatics*, 33(10): 1589–1590, 2017.

Bret E Peterson, Matthew D Healy, Prakash M Nadkarni, Perry L Miller, and Gordon M Shepherd. Modeldb: an environment for running and storing computational models and their results applied to neuroscience. *Journal of the American Medical Informatics Association*, 3(6):389–398, 1996.

Florian Prinz, Thomas Schlange, and Khusru Asadullah. Believe it or not: how much can we rely on published data on potential drug targets? *Nature reviews Drug discovery*, 10(9):712–712, 2011.

Kim D Pruitt, Tatiana Tatusova, and Donna R Maglott. Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic acids research*, 35(suppl_1):D61–D65, 2007.

Martin Przyjaciel-Zablocki, Alexander Schätzle, Thomas Hornung, and Georg Lausen. Rdfpath: Path query processing on large rdf graphs with mapreduce. In *Proceedings of the 2011 ESWC Workshops*, volume 7117, pages 50–64. Springer, 2012.

Manisha Pujari and Rushed Kanawati. Supervised rank aggregation approach for link prediction in complex networks. In *Proceedings of the 21st International Conference on World Wide Web*, pages 1189–1196. ACM, 2012.

Adrian Quintana, Matteo Cantarelli, Boris Marin, R Angus Silver, and Padraig Gleeson. Visualizing, editing and simulating neuronal models with the open source brain 3d explorer. *BMC neuroscience*, 16(1):P82, 2015.

Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases*. O'Reilly Media, CA, USA, 2013. ISBN 1449356265.

Nicolas Rodriguez, Jean-Baptiste Pettit, Piero Dalle Pezze, Lu Li, Arnaud Henry, Martijn P van Iersel, Gael Jalowicki, Martina Kutmon, Kedar N Natarajan, David Tolnay, et al. The systems biology format converter. *BMC bioinformatics*, 17(1): 154, 2016.

Hendrik Rohn, Astrid Junker, Anja Hartmann, Eva Grafahrend-Belau, Hendrik Treutler, Matthias Klapperstück, Tobias Czauderna, Christian Klukas, and Falk Schreiber. Vanted v2: a framework for systems biology applications. *BMC systems biology*, 6(1):139, 2012.

S. Rönnau et al. Towards xml version control of office documents. In *Proceedings of the 2005 ACM symposium on Document engineering*, pages 10–19. ACM, 2005.

S. Rönnau et al. Efficient and reliable merging of xml documents. In *Proceedings of the 18th ACM conference on Information and knowledge management (CIKM 09)*, pages 2105–2106. ACM, New York, NY, USA, 2009. doi: 10.1145/1645953.1646326.

Arévalo L. Rosado et al. An XQuery-based version extension of an XML Native Database. In *Proceedings of the 2009 EDBT/ICDT Workshops*, pages 99–106. ACM, 2009.

Robert Rosen. *Life itself: a comprehensive inquiry into the nature, origin, and fabrication of life*. Columbia University Press, 1991.

Christian Rosenke and Dagmar Waltemath. How can semantic annotations support the identification of network similarities. In *Proceedings of the 7th International Workshop on Semantic Web Applications and Tools for Life Sciences*, volume 1320, page 11, 2014.

Peter Saffrey and Richard Orton. Version control of pathway models using xml patches. *BMC systems biology*, 3(1):34, 2009.

G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):620, 1975.

Susanna-Assunta Sansone, Philippe Rocca-Serra, Dawn Field, Eamonn Maguire, Chris Taylor, Oliver Hofmann, Hong Fang, Steffen Neumann, Weida Tong, Linda Amaral-Zettler, et al. Toward interoperable bioscience data. *Nature genetics*, 44 (2):121, 2012.

Dewan M Sarwar, Reza Kalbasi, John H Gennari, Brian E Carlson, Maxwell L Neal, Bernard de Bono, Koray Atalag, Peter J Hunter, and David P Nickerson. Model annotation and discovery with the physiome model repository. *BMC bioinformatics*, 20(1):1–10, 2019.

Martin Scharm, Florian Wendland, Martin Peters, Markus Wolfien, Tom Theile, and Dagmar Waltemath. The CombineArchiveWeb application – A web based tool to handle files associated with modelling results. In *Proceedings of the 7th International SWAT4LS Workshop*, CEUR Workshop proceedings, 2014.

Martin Scharm, Olaf Wolkenhauer, and Dagmar Waltemath. An algorithm to detect and communicate the differences in computational models describing biological systems. *Bioinformatics*, 32(4):563–570, 2015.

Martin Scharm, Dagmar Waltemath, Pedro Mendes, and Olaf Wolkenhauer. Comodi: an ontology to characterise differences in versions of computational models in biology. *Journal of biomedical semantics*, 7(1):46, 2016.

Martin Scharm, Tom Gebhardt, Vasundra Touré, Andrea Bagnacani, Ali Salehzadeh-Yazdi, Olaf Wolkenhauer, and Dagmar Waltemath. Evolution of computational models in biomodels database and the physiome model repository. *BMC systems biology*, 12(1):53, 2018.

Rebekka Schlatter, Nicole Philippi, Gaby Wangorsch, Robert Pick, Oliver Sawodny, Christoph Borner, Jens Timmer, Michael Ederer, and Thomas Dandekar. Integration of boolean models exemplified on hepatocyte signal transduction. *Briefings in bioinformatics*, 13(3):365–376, 2011.

Falk Schreiber, Gary D Bader, Padraig Gleeson, Martin Golebiewski, Michael Hucka, Sarah M Keating, Nicolas Le Novère, Chris Myers, David Nickerson, Björn Sommer, et al. Specifications of standards in systems and synthetic biology: Status and developments in 2017. *Journal of integrative bioinformatics*, 15(1), 2018.

Marvin Schulz, Edda Klipp, Jannis Uhlendorf, and Wolfram Liebermeister. Sbmlmerge, a system for combining biochemical network models. *Genome Informatics*, 17(1):62–71, 2006.

Marvin Schulz, Falko Krause, Nicolas Le Novère, Edda Klipp, and Wolfram Liebermeister. Retrieval, alignment, and clustering of computational models based on semantic annotations. *Molecular systems biology*, 7(1), 2011.

Marvin Schulz, Edda Klipp, and Wolfram Liebermeister. Propagating semantic information in biochemical network models. *BMC bioinformatics*, 13(1):18, 2012.

Len Seligman and A Roenthal. Xml's impact an databases and data sharing. *Computer*, 34(6):59–67, 2001.

Bilal Shaikh, Lucian P Smith, Dan Vasilescu, Gnaneswara Marupilla, Michael Wilson, Eran Agmon, Henry Agnew, Steven S Andrews, Azraf Anwar, Moritz E Beber, et al. Biosimulators: a central registry of simulation engines and services for recommending specific tools. *Nucleic acids research*, 50(W1):W108–W114, 2022.

Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

Anatoly Sorokin, Nicolas Le Novère, Augustin Luna, Tobias Czauderna, Emek Demir, Robin Haw, Huaiyu Mi, Stuart Moodie, Falk Schreiber, and Alice Villéger. Systems biology graphical notation: entity relationship language level 1 version 2. *Journal of integrative bioinformatics*, 12(2):281–339, 2015.

Andrea Splendiani, Chris J Rawlings, Shao-Chih Kuo, Robert Stevens, and Phillip Lord. Lost in translation: data integration tools meet the semantic web (experiences from the ondex project). In *Recent Progress in Data Engineering and Internet Technology*, pages 87–97. Springer, 2012.

Natalie J Stanford, Martin Scharm, Paul D Dobson, Martin Golebiewski, Michael Hucka, Varun B Kothamachu, David Nickerson, Stuart Owen, Jürgen Pahle, Ulrike Wittig, et al. Data management in computational systems biology: exploring standards, tools, databases, and packaging best practices. In *Yeast Systems Biology*, pages 285–314. Springer, 2019.

Jakob Steiner. *Über die geometrischen Constructionen ausgeführt mittels der geraden Linie und eines festen Kreises.* Berlin, 1833.

Maciej Swat, Stuart Moodie, Niels Rode Kristensen, Nicolas Le Novère, et al. PharmML–An Exchange Standard for Models in Pharmacometrics: Version 0.2.1 (Specification), 2013.

Avraam Tapinos and Pedro Mendes. A method for comparing multivariate time series with different dimensions. *PloS one*, 8(2):e54201, 2013.

Chris F Taylor, Dawn Field, Susanna-Assunta Sansone, Jan Aerts, Rolf Apweiler, Michael Ashburner, Catherine A Ball, Pierre-Alain Binz, Molly Bogue, Tim Booth, et al. Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the mibbi project. *Nature biotechnology*, 26(8):889–896, 2008.

Mathialakan Thavappiragasam, Carol M Lushbough, and Etienne Z Gnimpieba. Automatic biosystems comparison using semantic and name similarity. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 790–796. ACM, 2014.

Ines Thiele, Neil Swainston, Ronan MT Fleming, Andreas Hoppe, Swagatika Sahoo, Maike K Aurich, Hulda Haraldsdottir, Monica L Mo, Ottar Rolfsson, Miranda D Stobbe, et al. A community-driven global reconstruction of human metabolism. *Nature biotechnology*, 31(5):419, 2013.

Krishna Tiwari, Sarubini Kananathan, Matthew G Roberts, Johannes P Meyer, Mohammad Umer Sharif Shohan, Ashley Xavier, Matthieu Maire, Ahmad Zyoud, Jinghao Men, Szeyi Ng, et al. Reproducibility in systems biology modelling. *Molecular Systems Biology*, 17(2):e9982, 2021.

Vasundra Touré, Alexander Mazein, Dagmar Waltemath, Irina Balaur, Mansoor Saqi, Ron Henkel, Johann Pellet, and Charles Auffray. Ston: exploring biological pathways using the sbgn standard and graph databases. *BMC bioinformatics*, 17 (1):494, 2016.

Vasundra Touré, Nicolas Le Novère, Dagmar Waltemath, and Olaf Wolkenhauer. Quick tips for creating effective and impactful biological pathways using the systems biology graphical notation. *PLoS computational biology*, 14(2):e1005740, 2018.

W. Tracz. Software reuse myths. *ACM SIGSOFT Software Engineering Notes*, 13 (1):17–21, 1988.

John J Tyson. Modeling the cell division cycle: cdc2 and cyclin interactions. *Proceedings of the National Academy of Sciences*, 88(16):7328–7332, 1991.

John J Tyson and Béla Novák. Functional motifs in biochemical reaction networks. *Annual review of physical chemistry*, 61:219–240, 2010.

Carel Van Gend, Riaan Conradie, Franco B Du Preez, and Jacky L Snoep. Data and model integration using jws online. *In silico biology*, 7(2 Supplement):27–35, 2007.

Martijn P Van Iersel, Alice C Villéger, Tobias Czauderna, Sarah E Boyd, Frank T Bergmann, Augustin Luna, Emek Demir, Anatoly Sorokin, Ugur Dogrusoz, Yukiko Matsuoka, et al. Software support for sbgn maps: Sbgn-ml and libsbgn. *Bioinformatics*, 28(15):2016–2021, 2012.

Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th annual Southeast regional conference*, page 42. ACM, 2010.

Dagmar Waltemath. Management of simulation studies in computational biology. *Invited presentations, junior research groups and research highlights at GCB*, 3: 1668, 2015.

Dagmar Waltemath and Olaf Wolkenhauer. How modeling standards, software, and initiatives support reproducibility in systems biology and systems medicine. *IEEE Transactions on Biomedical Engineering*, 63(10):1999–2006, 2016.

Dagmar Waltemath, Richard Adams, Daniel A Beard, Frank T Bergmann, Upinder S Bhalla, Randall Britten, Vijayalakshmi Chelliah, Michael T Cooling, Jonathan Cooper, Edmund J Crampin, et al. Minimum information about a simulation experiment (MIASE). *PLoS computational biology*, 7(4):e1001122, 2011a.

Dagmar Waltemath, Richard Adams, Frank T Bergmann, Michael Hucka, Fedor Kolpakov, Andrew K Miller, Ion I Moraru, David Nickerson, Sven Sahle, Jacky L Snoep, et al. Reproducible computational biology experiments with SED-ML – the simulation experiment description markup language. *BMC systems biology*, 5 (1):198, 2011b.

Dagmar Waltemath, Frank T. Bergmann, Richard Adams, and Nicolas Le Novère. Simulation Experiment Description Markup Language (SED-ML): Level 1 Version 1 (Specification), 2011c.

Dagmar Waltemath, Ron Henkel, Holger Meyer, and Andreas Heuer. Das Sombi-Framework zum Ermitteln geeigneter Suchfunktionen für biologische Modelldatenbasen. *Datenbank-Spektrum*, 11(1):27–36, 2011d.

Dagmar Waltemath, Neil Swainston, Allyson Lister, Frank Bergmann, Ron Henkel, Stefan Hoops, Michael Hucka, Nick Juty, Sarah Keating, Christian Knuepfer, et al. SBML Level 3 Package: Annot (Proposal), 2011e.

Dagmar Waltemath, Ron Henkel, Robert Hälke, Martin Scharm, and Olaf Wolkenhauer. Improving the reuse of computational models through version control. *Bioinformatics*, 29(6):742–748, 2013a.

Dagmar Waltemath, Ron Henkel, Felix Winter, and Olaf Wolkenhauer. Reproducibility of model-based results in systems biology. In *Systems Biology*, pages 301–320. Springer Netherlands, 2013b.

Dagmar Waltemath, Olaf Wolkenhauer, Nicolas Le Novère, and Michel Dumontier. Possibilities for integrating model-related data in computational biology. In

Proceedings of the 9th International Conference on Data Integration in the Life Sciences, Montreal, Canada, 2013c. CEUR Workshops.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, et al. Cord-19: The covid-19 open research dataset. *ArXiv*, 2020.

Rui-Sheng Wang, Assieh Saadatpour, and Reka Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Physical biology*, 9(5): 055001, 2012.

Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3, 2016.

Sarala M Wimalaratne, Matt DB Halstead, Catherine M Lloyd, Edmund J Crampin, and PF Nielsen. Biophysical annotation and representation of cellml models. *Bioinformatics*, 25(17):2263–2270, 2009.

Sarala M Wimalaratne, Pierre Grenon, Henning Hermjakob, Nicolas Le Novère, and Camille Laibe. Biomodels linked dataset. *BMC systems biology*, 8(1):91, 2014.

Sarala M Wimalaratne, Jerven Bolleman, Nick Juty, Toshiaki Katayama, Michel Dumontier, Nicole Redaschi, Nicolas Le Novère, Henning Hermjakob, and Camille Laibe. Sparql-enabled identifier conversion with identifiers.org. *Bioinformatics*, 31 (11):1875–1877, 2015.

U. Wittig et al. Sabio-rk: integration and curation of reaction kinetics data. In *Data integration in the life sciences*, pages 94–103. Springer, 2006.

Olaf Wolkenhauer. Why model? *Frontiers in physiology*, 5:21, 2014.

Katherine Wolstencroft, Stuart Owen, Olga Krebs, Wolfgang Mueller, Quyen Nguyen, Jacky L Snoep, and Carole Goble. Semantic data and models sharing in systems biology: The just enough results model and the seek platform. In *International Semantic Web Conference*, pages 212–227. Springer, 2013.

Katherine Wolstencroft, Stuart Owen, Olga Krebs, Quyen Nguyen, Natalie J Stanford, Martin Golebiewski, Andreas Weidemann, Meik Bittkowski, Lihua An, David Shockley, et al. Seek: a systems biology data and model management platform. *BMC systems biology*, 9(1):33, 2015.

Katherine Wolstencroft, Olga Krebs, Jacky L Snoep, Natalie J Stanford, Finn Bacall, Martin Golebiewski, Rostyk Kuzyakiv, Quyen Nguyen, Stuart Owen, Stian Soiland-Reyes, et al. Fairdomhub: a repository and collaboration environment for sharing systems biology research. *Nucleic acids research*, 45(D1):D404–D407, 2016.

Katy Wolstencroft, Stuart Owen, Matthew Horridge, Olga Krebs, Wolfgang Mueller, Jacky L Snoep, Franco du Preez, and Carole Goble. Rightfield: embedding ontology annotation in spreadsheets. *Bioinformatics*, 27(14):2021–2022, 2011.

Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 721–724. IEEE, 2002.

Tommy Yu, Catherine M Lloyd, David P Nickerson, Michael T Cooling, Andrew K Miller, Alan Garny, Jonna R Terkildsen, James Lawson, Randall D Britten, Peter J Hunter, et al. The physiome model repository 2. *Bioinformatics*, 27(5):743–744, 2011.

Tommy Yu, Randall D Britten, David P Nickerson, Peter Hunter, and Poul MF Nielsen. Physiome repository. *Encyclopedia of Computational Neuroscience*, pages 2405–2406, 2015.

Bingjun Zhang, Jialie Shen, Qiaoliang Xiang, and Ye Wang. Compositemap: a novel framework for music similarity measure. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 403–410, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1572011.

Fengkai Zhang, Lucian P Smith, Michael L Blinov, James Faeder, William S Hlavacek, Jose Juan Tapia, Sarah M Keating, Nicolas Rodriguez, Andreas Dräger, Leonard A Harris, et al. Systems biology markup language (sbml) level 3 package: multistate, multicomponent and multicompartment species, version 1, release 2. *Journal of Integrative Bioinformatics*, 17(2-3), 2020.

A. Appendix

A.1. Publications

Journal (peer-reviewed)

Henkel, R., Endler, L., Peters, A., Le Novère, N. & Waltemath, D. (2010). Ranked retrieval of computational biology models. BMC Bioinformatics, 11(1), 423.

Waltemath, D., Henkel, R., Meyer, H. & Heuer, A. (2011). Das Sombi-Framework zum Ermitteln geeigneter Suchfunktionen für biologische Modelldatenbasen. Datenbank-Spektrum, 11(1), 27-36.

Waltemath, D., Henkel, R., Hälke, R., Scharm, M. & Wolkenhauer, O. (2013). Improving the reuse of computational models through version control. Bioinformatics, 29(6), 742-748.

Henkel, R., Wolkenhauer, O. & Waltemath, D. (2015). Combining computational models, semantic annotations and simulation experiments in a graph database. Database, 2015, bau130.

Alm, R., Waltemath, D., Wolfien, M., Wolkenhauer, O. & Henkel, R. (2015). Annotation-based feature extraction from sets of SBML models. Journal of Biomedical Semantics, 6(1), 20.

Henkel, R., Hoehndorf, R., Kacprowski, T., Knüpfer, C., Liebermeister, W. & Waltemath, D. (2016). *Notions of similarity for systems biology models.* Briefings in Bioinformatics, bbw090.

Touré, V., Mazein, A., Waltemath, D., Balaur, I., Saqi, M., Henkel, R., Pellet, J. & Auffray, C. (2016). STON: exploring biological pathways using the SBGN standard and graph databases. BMC Bioinformatics, 17(1), 494.

Yordanova, K., Koldrack, P., Heine, C., Henkel, R., Martin, M., Teipel, S. & Kirste, T. (2017) Situation Model for Situation-Aware Assistance of Dementia Patients in Outdoor Mobility. IOS Press, 1461 – 1476.

Keating, S., Waltemath, D., König, M., Zhang, F., Dräger, A., Chaouiya, C., Bergmann, F., Finney, A., Gillespie, C., Helikar, T., Henkel, R., and others. (2020). SBML Level 3: an extensible format for the exchange and reuse of biological models. Molecular Systems Biology, 16 (8)

Gütebier, L., Bleimehl, T., Henkel, R., Munro, J., Müller, S., Morgner, A., Laenge, J., Pachauer, A., Erdl, A., Weimar, J., Walther-Langendorf, K., Vialard, V., Liebig, T., Preusse, M. Walthemath, D. & Jarasch, A. (2022). CovidGraph: A Graph to fight COVID-19. Bioinformatics. 38 (20), btac592

Journal (not peer-reviewed)

Waltemath, D., Swainston, N., Lister, A. L., Bergmann, F., Henkel, R., Hoops, S., Hucka, M., Juty, N., Keating, S., Knuepfer, C., Krause, F., Laibe, C., Liebermeister, W., Lloyd, C., Goksel, M., Schulz, M., Taschuk, M. & Le Novère, N. (2011). SBML level 3 package proposal: annotation.

Henkel, R., Lambusch, F., Wolkenhauer, O., Sandkuhl, K., Rosenke, C. & Waltemath, D. (2016). Finding patterns in biochemical reaction networks. PeerJ PrePrints, 4, e1479v2.

Book chapter

Waltemath, D., Henkel, R., Winter, F. & Wolkenhauer, O. (2013). Reproducibility of model-based results in systems biology. In Systems Biology (pp. 301-320). Springer Netherlands.

Lange, M., Henkel, R., Müller, W., Waltemath, D., & Weise, S. (2014). Information retrieval in life sciences: a programmatic survey. In Approaches in Integrative Bioinformatics (pp. 73-109). Springer Berlin Heidelberg.

Henkel, R., Waltemath, D. (2020). Biomedical repositories for simulation studies. In Systems Medicine: Integrative, Qualitative and Computational Approaches. Elsevier Netherlands.

Conferences

Köhn, D., Maus C., Henkel, R., Kolbe, M. (2009) Towards enhanced retrieval of biological models through annotation-based ranking. In: 6th International Workshop Data Integration in the Life Sciences (DILS 2009), Manchester, UK.

Henkel, R., Köhn, D., Peters, A. & Maus, C. (2009). Similarity Measures for Computational Biological Models. In: 6th International Workshop Data Integration in the Life Sciences (DILS 2009), Manchester, UK.

Henkel, R., Le Novère, N., Wolkenhauer, O. & Waltemath, D. (2012). Considerations of graph-based concepts to manage of computational biology models and associated simulations. GI-Jahrestagung 2012 (pp. 1545–51)

Henkel, R. & Waltemath, D. (2014). MaSyMoS: Finding Hidden Treasures in Model Repositories. In SWAT4LS.

Koldrack, P., Henkel, R., Krüger, F., Teipel, S. & Kirste, T. (2015). Supporting situation awareness of dementia patients in outdoor environments. In Proceedings of the 9th International Conference on Pervasive Computing Technologies for Healthcare (pp. 245-248). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Koldrack, P., Henkel, R., Zarm, K., Teipel, S. & Kirste, T. (2015). Situationsadaptive Navigationsassistenz für Menschen mit Demenz. In AAL-Kongress 2015. VDE VERLAG GmbH.

Gütebier, L., Henkel, R., Jarasch, A., Bleimehl, T. Müller, S., Munro, J., Preusse, M., Walthemath, D. & The HealthEcco Team. (2021). CovidGraph: Connecting biomedical COVID-19 resources and computational biology models. 2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores, SEA-Data 2021. Workshop at 47th International Conference on Very Large Data Bases, 2021.

Gütebier, L., Waltemath, D. & Henkel, R. (2022). Data Exploration with CovidGraph. 32nd Medical Informatics Europe Conference.

Gütebier, L., Bleimehl T., Henkel, R., Preusse M. & Waltemath D. (2022) The HealthECCO knowledge graph: applying Neo4j technologies in health data research. NODES2022. Nov. 17th 2022

A.2. Publication Acknowledgments

Listed here are chapters, sections and subsections that are based on or are adapted from self authored publications. For each chapter, section or subsection the relevant publications are provided. All chapters, sections and subsections listed here may contain literal quotes or may be considered as a (partially) literal quote of their respective publication.

Also listed are the acknowledgements, funding declarations and author contributions for the four main publications described in the chapters 4 to 7.

A models habitat

The Chapter 2 is based on several publications. For this Chapter the following publications have been adapted: Sections 2.1, 2.2.1, and 2.2.2 from Waltemath et al. [2013b]. Section 2.2.1, Paragraph SBGN from Touré et al. [2016]. Section 2.3.1 from Henkel [2020]. Section 2.4 from Henkel [2009].

Ranked retrieval of computational biology models

The Chapter 4 is based on the publication Henkel et al. [2010]. The application of ranking and retrieval methods on bio-models based on model annotations was suggested by Dagmar Waltemath (DW). Andre Peters (AP), Ron Henkel (RH) and DW discussed different similarity functions. RH developed and implemented the approach in BioModels Database, supervised by Nicolas le Novère (NLN). Lukas Endler (LE) and RH discussed and determined the different weights for features and qualifiers used for the similarity function. LE provided detailed examples for the evaluation of the approach. All authors contributed to the manuscript.

The Authors RH and DW were supported by the German Research Association (DFG) research training group dIEM oSiRiS (DFG grant 1387). Implementation work at the EBI, done by RH, was funded by the Leonardo da Vinci - European Commission's Lifelong Learning Programme. AP would like to thank for support through the DFG research training group MUSAMA (DFG grant 1424). The development of BioModels Database is funded by the European Molecular Biology Laboratory, the Biotechnology and Biological Science (grant BB/F010516/1), and the National Institute of General Medical Sciences (grant R01 GM070923). Authors are grateful to Camille Laibe and the BioModels.net Team for their help with the implementation in BioModels Database.

Model version control - Improving the reuse of computational models through version control

The Chapter 5 is based on the publication Waltemath et al. [2013a].

Dagmar Waltemath (DW), Ron Henkel (RH) and Olaf Wolkenhauer (OW) identified the requirements for model version control. DW and RH developed the concepts. Robert Hälke (RHä) originally implemented the *BiVeS* library, supervised by RH. MS extended *BiVeS* and implemented the online application *BudHat*. This work was funded by the German Federal Ministry of Education and Research (e:Bio, SEMS). The authors would like to thank the COMBINE community for valuable discussions on model version control during the COMBINE 2012 meeting.

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

The Chapter 6 is based on the publications Henkel et al. [2012b] and Henkel et al. [2015].

Ron Henkel (RH) and Dagmar Waltemath (DW) designed the project and wrote the paper. RH implemented the storage system. Olaf Wolkenhauer (OW) reviewed paper drafts and supervised the project.

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. The authors also express their gratitude to the reviewers who provided very detailed and helpful comments. The authors acknowledge financial support from the BMBF (Junior Research Group SEMS, e:Bio program, grant no. FKZ0316194).

Notions of similarity for systems biology models

The Chapter 7 is based on the publication Henkel et al. [2016a]. The Authors contributed equally to the paper.

The authors thank Wolfgang Müller, Matthias Lange and Martin Scharm for valuable discussions on similarity notions of systems biology models. This article was drafted during a meeting that was organized by Dagmar Waltemath (DW) and funded through the BMBF e:Bio program (grant no. FKZ0316194). Ron Henkel is funded by the German Federal Ministry of Education and Research (BMBF; grant number FKZ 031 A540A [de.NBI]). The Junior Research Group SEMS, BMBF e:Bio program (grant no. FKZ0316194 to DW). Tim Kacprowski is funded by the German

Federal Ministry of Education and Research (BMBF) via the Greifswald Approach to Individualized Medicine (GANI_MED; grant 03IS2061A) and by the Unternehmen Region as part of the ZIK-FunGene (grant 03Z1CN22). German Research Foundation (grant no. Ll 1676/2-1 to Wolfram Liebermeister).

Habitat status report

The Chapter 8 describes the current state-of-the-art and is intended to be an update for the information provided in chapters 4 to 7. For this Chapter the following publications have been adapted: Subsection 8.1.4 is based on Nassar et al. [2015]. Subsection 8.3.1 is based on two publications, Alm et al. [2014] and Alm et al. [2015]. Subsection 8.3.2 is based on Henkel et al. [2016b] and Lambusch et al. [2018]. Section 8.4 is based on Touré et al. [2016]. Section 8.5 is based on Gütebier et al. [2021] and Gütebier et al. [2022].

B. Summary

B.1. Dissertation Summary

This thesis addresses reproducibility and reusability of systems biology models and elaborates concepts of model management to support a FAIR (findable, accessible, interoperable, reusable) handling of systems biology models.

To this end, this thesis focuses on six key research questions:

1. How to establish a baseline for model management?
2. How can heterogeneous, semi-structured model data be retrieved and ranked?
3. How to trace a model over its lifetime?
4. How to store a model and its accompanying meta-data?
5. What model parts are important to elaborate on model similarity?
6. What becomes possible once the aforementioned research questions are answered?

To answer these questions, this thesis describes the development of concepts and implementations for model storage, model search, model version control, and model similarity analysis. These developments are put in perspective regarding the state of the art at the time of development and regarding the current state of the art, respectively. In summary, this thesis provides an overview of the means and measures of model management and describes in its individual chapters solutions for various aspects of a FAIR model management. Finally, the approaches presented here are taken up and extended in the context of the current COVID-19 research.

B.2. Zusammenfassung der Dissertation

Die hier vorliegende Arbeit beschäftigt sich mit der Reproduzierbarkeit und Wiederverwendbarkeit systembiologischer Modelle. Die Arbeit behandelt Konzepte des Modellmanagements um eine FAIRe (auffindbar, zugreifbar, interoperabel, wiederverwendbar) Handhabung von systembiologischen Modellen zu unterstützen.

Im Fokus dieser Arbeit stehen sechs Forschungsfragen:

1. Welche Maßnahmen sind nötig um eine Grundlage für Modellmanagement zu etablieren?
2. Wie können heterogene, semi-strukturierte Modelle mittels Suchanfrage gefunden und bewertet werden?
3. Wie kann die Weiterenticklung eines Modells über die Zeit verfolgt werden?
4. Wie können, neben dem Modell, auch die zugehörigen Metadaten gespeichert werden?
5. Welche Modellkomponenten sind wichtig um eine Aussage über Modellähnlichkeit treffen zu können?
6. Welche weiteren Optionen werden verfügbar, wenn Konzepte und Technologien zur Beantwortung der vorherigen Fragen existieren?

Zur Beantwortung dieser Fragen werden in dieser Arbeit Konzepte und Umsetzungen zur Modellspeicherung, Modellsuche, Modellversionierung und zur Bewertung von Modellähnlichkeiten entwickelt. Diese Entwickelungen werden in den Wissensstand zur Entwicklungszeit und vergleichend zum aktuellen Wissensstand eingeordnet. Zusammenfassend vermittelt diese Arbeit somit ein Bild des aktuellen Standes des Modellmanagements und beschreibt in ihren einzelnen Kapiteln Lösungsmöglichkeiten für verschiedene Aspekte eines FAIRen Modellmanagements. Abschließend werden die hier vorgestellten Ansätze aufgegriffen und im Kontext der aktuellen COVID-19 Forschung erweitert.

C. Acknowledgements

> "Your path led through the sea, your way through the mighty waters, though your footprints were not seen." **Psalm 77:19**

The well known bible metaphor "Footprints in the Sand" also holds true for writing a PhD thesis. As this thesis spans a 12 year time period of research, the list of people I got to know, worked with, cooperated with, was influenced by, got help from, discussed with, argued with, [...], loved and lost would fill an entire chapter. I am thankful for every single person on this list - you never walk alone! However, there are people in my life who deserve a special thank you.

Firstly, I want to thank my parents - for everything the have done for me and for everything they are sill doing for me, for having supported me my entire life, raising me, teaching me, friendly pushing me whenever necessary and always loving me.

Secondly, I want to thank Dagmar Waltemath - for 14 years of scientific collaboration, working together and for her friendship. Thank you for providing the topic and supervising my diploma thesis, which kind of sparked my interest in the field of model management. Thank you for your continues supervision and support, for our fruitful discussions, and for always being welcome to speak openly and freely no matter what.

Last but not least, I want to thank Kurt Sandkuhl - for taking me in when I was a lost PhD student. Thank you for your support and supervision, for disassembling and reassembling my research in a better way, for guiding me methodically and for your unbelievable patience during my slow thesis writing process.

D. Selbständigkeitserklärung

Ich versichere eidesstattlich durch eigenhändige Unterschrift, dass ich die Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht und ist in gleicher oder ähnlicher Weise noch nicht als Promotionsschrift zur Anerkennung oder Bewertung vorgelegt worden.

Ort, Datum	Ron Henkel

Lebenslauf

STAMMDATEN

Name:	Ron Henkel
Staatsangehörigkeit:	deutsch
Geburtsdatum:	19.09.1982
Geburtsort:	Waren (Müritz)
Familienstand:	ledig

AUSBILDUNG UND QUALIFIKATION

seit Dezember 2009	Doktorand mit dem Forschungsgebiet Datenintegration und Datenmanagement, relationale und graphbasierte Datenbanken, Information Retrieval, Systembiologie; *Universität Rostock*
September 2009	Abschluss Diplom-Informatik (Note: sehr gut); *Universität Rostock*
Oktober 2003 - September 2009	Studiengang Diplom-Informatik; Vertiefung: Datenbanken und Informationssysteme, Nebenfach: Biologie; *Universität Rostock*
Juni 2002	Abitur (Note: gut); *Richard-Wossidlo-Gymnasium Waren*

ARBEITSERFAHRUNG

seit Oktober 2019	Wissenschaftlicher Mitarbeiter und stellvertetender Abteilungsleiter Medizininformatik am Uniklinikum Greifswald. Verantwortlich für Forschung und Lehre im Bereich der medizinischen Informatik, speziell Datenintegration und Datenspeicherung. *Abt. Medizininformatik, Institut für Community Medicine, Universitätsmedizin Greifswald*
April 2018 - September 2019	Leiter des IT-Dezernats am LAGuS. Verantwortlich für sechs Mitarbeiter. Bisherige Schwerpunkte im Bereich der Sicherstellung des IT-Supports, Einführung der E-Akte und von Verfahren nach dem Onlinezugangsgesetz, Lizenzmanagement, Weiterentwicklung und Durchsetzung der IT-Sicherheitsrichtlinie, Verfassen von Verfahrensbeschreibungen nach DSGVO, Budgetplanung und die Koordination externer Dienstleister. *Dezernent Informationstechnik, Landesamt für Gesundheit und Soziales*
September 2017 - März 2018	Leiter des IT-Dezernats am LUNG (Vertretung für Mutterschutz und Elternzeit). Verantwortlich für acht Mitarbeiter. Schwerpunkte im Bereich der Sicherstellung des IT-Supports, Lizenzmanagement, Weiterentwicklung und Durchsetzung der IT-Sicherheitsrichtlinie, Verfassen von Verfahrensbeschreibungen nach §18 DSG M-V, Budgetplanung für die restlichen Haushaltsmittel sowie die Koordination externer Dienstleister. *Dezernent Informationstechnik und Umweltinformationssystem, Landesamt für Umwelt, Naturschutz und Geologie*
Juni 2015 - Dezember 2017	Projektarbeit im Rahmen von de.NBI und Elixir mit den Schwerpunkten der Integration einer Modellsuche in ein Daten- und Modellmanagementsystem, Ausbildung von Systembiologen und Bioinformatikern im Daten- und Modellmanagement (Europa & USA). Mitarbeit in der Arbeitsgruppe Infrastructure & Data Management. *Scientific Databases and Visualization, Heidelberg Institut für Theoretische Studien*
August 2014 - Juni 2015	Wissenschaftlicher Mitarbeiter (Projektstelle SiNDeM) im Forschungsthema Situationsadaptive Navigationsassistenz für Menschen mit Demenz. Schwerpunkte des Projektes war die Konzeption, Ausführung und Auswertung von Experimenten zur Erhebung von Verhaltens- und Bewegungsdaten von Menschen mit Demenz in städtischer Umgegung. Mitwirkung an Forschunganträgen, Erstellen einer Verfahrensbeschreibung. *Lehrstuhl Mobile Multimedia Information Systems, Universität Rostock*
März 2013 - April 2013	Forschungsaufenthalt am Auckland Bioengineering Institute zur Integration einer Modellsuche in das Modellmanagementsystem PMR2. *Auckland Bioengineering Institute (ABI), Auckland, Neuseeland*
Mai 2011 - Mai 2014	Wissenschaftlicher Mitarbeiter (Haushaltsstelle) mit dem Forschungsthema Konzeption und Entwicklung eines graphbasierten Managementsystems zur Verwaltung systembiologischer Modelle und Simulationsbeschreibungen. Mitwirkung in der Lehrstuhlverwaltung, Koordinator für Lehraufgaben am Lehrstuhl (ab 2012) und Mitwirkung an Forschungsanträgen. *Lehrstuhl Systembiologie, Universität Rostock*
Mai 2011 - Juni 2017	Dozent für die Veranstaltungen Abstrakte Datentypen und Datenstrukturen, Datenbanken und Markupsprachen in der Systembiologie, Methoden der Wirtschaftsinformatik. *Universität Rostock*

ARBEITSERFAHRUNG (FORTSETZUNG)

Januar 2010 - April 2011	Stipendiat im interdisziplinären Graduiertenkolleg zum Forschungsthema Konzeption von Verfahren zur relationalen Speicherung und Suche von systembiologischen Modellen. Mitwirkung an Forschungsanträgen. *Lehrstuhl Datenbank- und Informationssysteme, Universität Rostock*
September 2009 - Dezember 2009	Forschungspraktikum in der Computational Systems Neurobiology Group zur Entwicklung und Implementierung einer Modellsuche für das Modellmanagementsystem BioModels Database. *European Bioinformatics Institute (EMBL-EBI), Cambridge (Hinxton), UK*
seit September 2009	Eigenständige Betreuung wissenschaftlicher Abschlussarbeiten und Projekte Studierender der Informatik und Wirtschaftinformatik. *Universität Rostock*
April 2008 - Oktober 2008	Praktikum als Entwickler für IBM InfoSphere MashupHub. *IBM, San José, USA*
Oktober 2007 - März 2008	Praktikum als Entwickler für zSeries Firmware Development Support. *IBM Deutschland Research & Development, Böblingen*

AUSZEICHNUNGEN

Februar 2013	Hermes Travel Award für das Projekt Information Retrieval im Physiome Model Repository. *Universität Rostock*
Januar 2009 - September 2009	Diplomarbeit: „Determining Model Similarities through Ranking Functions using the Example of Biological Computational Models“ Note 1,0; ausgezeichnet mit dem Preis INFO.RO 2009 als beste Diplomarbeit des Instituts für Informatik 2009. *Universität Rostock*

SPRACHEN

Deutsch	Muttersprache
Englisch	Verhandlungssicher
Russisch	Grundkenntnisse