Annotation-based storage and retrieval of models and simulation descriptions in computational biology

Dissertation
zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

Promotionsgebiet: Datenbank- und Informationssysteme

vorgelegt von
Dagmar Waltemath geb. Köhn,
wohnhaft in Rostock,
geboren am 23.03.1981 in Waren (Müritz)

Universität Rostock Traditio et Innovatio

Rostock, 23. 03. 2011

urn:nbn:de:gbv:28-diss2011-0134-5

Principal advisor:	**Prof. Dr. rer. nat. habil. Andreas Heuer**, Database and Information Systems, University of Rostock, Rostock, Germany
External reviewers:	**Dr. Nicolas Le Novère**, European Bioinformatics Institute, Cambridge, UK
	Prof. Dr. Ulf Leser, Institute for Computer Science, Humboldt-Universität zu Berlin, Berlin, Germany
Date of defense:	2011-07-27
URN:	urn:nbn:de:gbv:28-diss2011-0134-5

Abstract

The purpose of this work was to enhance the reuse of computational biology models by identifying and formalizing relevant meta-information, and to develop extended model storage and retrieval concepts.

The work focussed on models in XML formats and annotated with RDF-encoded meta-information. One particular type of meta-information investigated in this thesis is experiment-related meta-information attached to a model, which is necessary to accurately recreate simulations. The implementations in this work were carried out using methods from the research area of 'Information Retrieval', and proposed solutions were tested on the BioModels Database repository.

The main results of the study are: a detailed concept for model annotation, a proposed format for the encoding of simulation experiment setups, a storage solution for standardized model representations, and the development of a retrieval concept.

A model annotation strategy has been developed for the model representation format SBML. The proposed simulation experiment encoding is being considered by various groups within the computational biology arena, including the SBML, CellML and NeuroML working groups. It has been shown that a fine-grained storage approach enables model search and versioning across different model representation formats. The application of information retrieval techniques indicated how the model search process can be enhanced by considering meta-information and incorporating information from external resources.

The principal conclusion was that a fine-grained encoding and storage of model meta-information enhances the retrieval process and thereby improves model reuse.

Zusammenfassung

Die vorliegende Arbeit widmete sich der besseren Wiederverwendung biologischer Simulationsmodelle. Ein Ziel war dabei die Identifikation und Formalisierung relevanter Modell-Meta-Informationen. Ein weiterer Schwerpunkt lag in der Entwicklung geeigneter Modellspeicherungs- und Modellretrieval-Konzepte.

Fokus der Arbeit waren in XML kodierte Modelle, die mit in RDF gespeicherten Meta-Informationen angereichert wurden. Besondere Beachtung fand hierbei Meta-Information über Experimente auf einem Modell. Realisiert wurde das Modellretrieval mit Methoden des Information-Retrievals. Die vorgeschlagene Lösung wurde in dem Repository 'BioModels Database' implementiert.

Wichtigste Ergebnisse der Arbeit sind ein detailliertes Konzept zur Modellannotation, ein Formatvorschlag für standardisierte Kodierung von Simulationsexperimenten in XML, eine Speicherlösung für Modellrepräsentationen, sowie ein Retrieval-Konzept.

Im Rahmen der Dissertation wurde das Konzept der Modellannotationen für das Repräsentationsformat SBML überarbeitet. Das entwickelte Format für Simulationsexperimentspeicherung wird in verschiedenen Gruppen getestet, z. B. SBML, CellML und NeuroML. Das feingranulare Speicherungskonzept für Simulationsmodelle realisiert eine formatunabhängige Modellsuche und -versionierung. Der Einsatz von Information-Retrieval-Techniken zeigt, wie Modellsuche verbessert werden kann, wenn zsätzlich zur Modellstruktur Meta-Informationen und externe Resourcen einbezogen werden.

Zusammenfassend kann festgehalten werden, dass eine feingliedrige Kodierung und Spei-cherung von Modell-Meta-Informationen das Retrieval verbessert und somit einen positiven Effekt auf die Modellwiederverwendung hat.

It was still possible in the 1960s for a human living to live in a nation, and be subject to its laws, without the slightest knowledge of that fact. If we find this astonishing, it is because we human beings, unlike all other species on the planet, are knowers. We are the only ones who have figured out what we are, and where we are, in this great universe. And we're even beginning to figure out how we got there.

(D. C. Dennett, Freedom Evolves [Dennett 2004, Chap. 1])

Acknowledgements

Nicht, weil es so schwer ist, wagen wir es nicht, sondern, weil wir es nicht wagen, ist es so schwer.

(Lucius Annaeus Seneca)

It is my pleasure to thank the different people who have made this thesis possible.
First of all, I would like to thank professor Adelinde Uhrmacher and professor Andreas Heuer for giving me the opportunity to join the dIEM oSiRiS research training school. Being part of an international and integrative project was an exciting and invaluable experience for me. Thanks also goes to my thesis committee for many helpful comments on my work. Thank you for taking the time and effort to read my thesis and traveling to Rostock for the thesis defense.

I owe my deepest gratitude to Dr. Nicolas Le Novère for his continuous support throughout my phd studies – for his enthusiasm for my research work, for giving me the opportunity to present my work at conferences and get in touch with the relevant people in the field, for his patience in teaching me how to write "proper scientific articles", for his advises on the various aspects of a phd student's life in general. I am also grateful to professor Olaf Wolkenhauer for continuous help and support, in particular for having me join the group's research seminar and feedback on my work.

Most of my work was "community based" and I would therefore like to take this opportunity to thank my colleagues in the SBML and SED-ML community for endless scientific discussions and many exciting meetings. My work would be nothing without you.

A phd thesis cannot be done without the support of the likely-minded, also suffering phd colleagues. Thank you, RIGZ-colleagues for an enjoyable time in the Rostock office, for making me become a coffee-addict, for wonderful Christmas parties, BBQs, wine deliveries and baby-BATS. I am grateful to my colleagues at CompNeur for teaching me how diverse life in a research group can be. Thank you for making me an expert in table football, badminton, piano playing, jogging and bus-pub-hopping. I would also like to thank my colleagues from the SBI group who so kindly welcomed me in their group. Your coffee is the best I have ever had. Thank you, Yvonne, Felix and Ron for doing all the printing jobs for me! I am pleased to thank Ron Henkel, who has contributed a lot to this thesis in many ways. Finally, I would like to thank Lu Li for encouraging me not to give up, without her I might have stopped and rewinded.

This thesis could not have been written without the support of my family and friends, in particular without the various baby sitters who took care of Emil & Hans during my working hours. Special thanks goes to my parents and my parents-in-law, but also to Yvonne for organising pyjama parties for my elder son, to Ron and Katja who could not be scared away by bacillus *Bacillicus*, to Virginia who sang good night songs to my younger son in the office. You were great.

At the very last, I need to thank the *Deutsche Bahn* for enjoyable traveling days in which I was cut-off the World Wide Web. Those were the days I worked most efficiently.

Dagmar Waltemath
Rostock, September 19, 2011

This research has been financially supported by the interdisciplinary DFG Graduiertenkolleg dIEM oSiRiS. Research visits to the European Bioinformatics Institute were kindly funded by the Marie-Curie Fellowship and the DFG.

Contents

List of Figures

List of Tables

1. Introduction

The re-use of models requires an understanding for the information that is needed to support reuse and how it should be presented [..].

(Sauro et al. [2006], citing [Overstreet et al. 2002])

Computer science technologies and methodologies nowadays support research in various areas. As one application, they help understanding biological systems by means of modeling and simulation techniques. This accelerates the modeling and simulation process, and allows for sophisticated analyses of complex biological systems. Systems biologists study biological systems such as pathways inside a cell, cell interactions, or whole organs. Testing hypotheses computationally facilitates reduction of time and costs for experimental biologists.

The modeler relies on previously obtained findings from collaborators and literature for a modeling project to work. The evolution of the *cell cycle model* which investigates the process a cell undergoes to divide and replicate itself is an example for the reuse of existing knowledge. The cell is a so-called autopoietic system, i. e. it is capable of self-creating the processes that reproduce itself. One key process in cell replication is the division of the cell. Cell division periodically appears in unicellular systems. A well-known representation of that phenomenon has been described by John Tyson in a small mathematical model in 1991 [Tyson 1991]. The publication explains the complex cell cycle mechanism in a mathematical model using only six interacting species and nine reactions. Tyson showed that the oscillating concentration of the Maturation Promotion Factor (*MPF*) is dependent on the cyclin concentration. In the following years, Tyson's lab developed a number of enhanced models based on the first representations of the cell cycle.

A computational model of a biological system is an abstract representation of the living system, simplified by a number of restrictions, and instantiated with a certain set of parameter values. Computational models are typically encoded in specifically designed, computer-readable formats – allowing for model exchange across different software tools. One way of studying the models is through simulation. A simulation mimics the temporal development of the system, for example, determining the changes in concentration of a particular entity over time.

The model description solely represents model structure and initial parametrization. It lacks formal descriptions of model-related information such as the build-in assumptions, the modeler's intention, and related simulations on the considered model. However, these are important aspects of a model – not only to understand the model and to allow for reuse in other contexts, but also to verify the results. Looking at the first Tyson model, one learns a lot about the cell cycle. However, the model misses information on the implicitly made assumptions and also on the scope of the modeled system.

Incorporating additional information gives further insights (e. g. reading the reference publication, studying related models, or investigating simulation experiments performed on the model). If, in addition, the model is sufficiently described by meta-information, one can infer the considered aspects and implied assumptions. For example, a model linked to a particular biological system (such as "unicellular biological systems") will be more applicable and reusable; 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 understood, and modifications on the simulation settings can be easier reconstructed.

Given that additional information for a model can be provided, computational support is needed to facilitate efficient storage, versioning, and later retrieval of both the model and its associated information.

In this thesis, I argue that improved reuse of computational biology models can only be achieved with an enhanced understanding of the model and its components. The understanding depends on the available information on the model. Relevant models can be applied to the modelers problem with less effort when providing detailed information on the model entities, the assumptions, the model context and behavior, and also the simulations. The computer-interpretable encoding of that information enables support by computational methods. Appropriate storage and search techniques have to be developed to realize successful model retrieval. With the growing number of existing models, the development of ranking mechanisms that suggest a user the best model for his needs has become necessary. Solutions are proposed and discussed in this work.

1.1. Context

The research that led to this thesis was funded by the research training school "**d**ie **I**ntegrative **E**ntwicklung von **Mo**dellierungs- und **Si**mulationsmethoden für **R**egenerat**i**ve **S**ysteme" (The Integrative Development of Modeling and Simula-

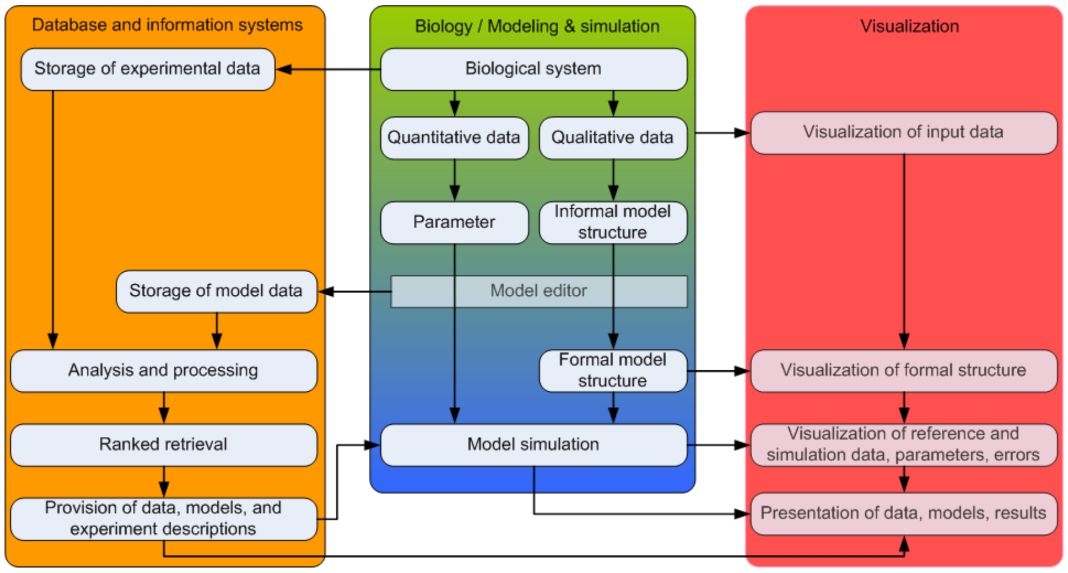

Database and information systems
Storage of experimental data
Storage of model data
Analysis and processing
Ranked retrieval
Provision of data, models, and experiment descriptions
Biology / Modeling & simulation
Biological system
Quantitative data
Qualitative data
Parameter
Informal model structure
Model editor
Formal model structure
Model simulation
Visualization
Visualization of input data
Visualization of formal structure
Visualization of reference and simulation data, parameters, errors
Presentation of data, models, results

Figure 1.1.: The systems biology workflow from the dIEM oSiRiS perspective (green-blue box), the challenges for visualization (red box) and for database and information systems (orange box) research. Extended from [Unger et al. 2007].

tion Methods for Regenerative Systems) (dIEM oSiRiS, [Uhrmacher et al. 2009]). dIEM oSiRiS is a joint project of different groups at the Institute of Informatics and the Faculty of Medicine and Biology at Rostock University (Germany). Within the research training school, Biologists perform experiments and generate experimental data that is then used in the dry laboratory (i. e. computationally) to build models and to evaluate newly developed modeling and simulation techniques. The graduate school also elaborates on visualization techniques for the different kinds of data. Unger et al. [2007] describes the Systems Biology workflow from biological data to the simulation of a computational model from the perspective of the dIEM oSiRiS project. Since the description only reflects a data-oriented view, Figure 1.1 shows a revised version of that workflow, integrating also database aspects.

The box in the middle (blue-green) represents the dIEM oSiRiS workflow for the modeling of biological systems. The dissertation of Unger [2010] discusses missing parts of the original workflow from a visualization tools' perspective, meaning the evaluation of visualization support (red box). Contrarily, my work discusses missing support on the database and information systems side (orange box). The demands are not restricted to the dIEM oSiRiS project, but apply to computational biology in general.

The dIEM oSiRiS project studies new methods for the modeling of biological systems which are explored in the biology laboratory at the same time (green box). The complex functioning of biological systems cannot be understood by pure intuition [Kitano 2002b; Klipp et al. 2009]. Focusing only on the individual parts of a biological system (e. g. its individual proteins or organs) will miss on properties of the system that emerge only in the network of interactions that is formed by its components [Nature 2005]. *Systems Biology* investigates the system-level understanding of biology [Kitano 2001, 2002b], making it an "integrative study of the interactions between different components of biological systems, and how such interactions give rise to the function and behavior of a system" [De Schutter 2008]. The *EraNet*[1], a consortium of funding bodies for systems approaches to biological research, agreed to define the systems biology approach as follows:

> Systems biology aims at understanding the dynamic interactions between components of a living system, between living systems and their interaction with the environment. Systems biology is an approach by which biological questions are addressed through integrating experiments in iterative cycles with computational modelling, simulation and theory. Modelling is not the final goal, but is a tool to increase understanding of the system, to develop more directed experiments and finally allow predictions. [Pastori et al. 2008]

Biology experiments lead to hypotheses about a system, but these hypotheses often cannot be combined into a larger picture as the global behavior cannot be inferred from the knowledge of its parts [Klipp et al. 2009, pp. 4-5]. In this situation, *mathematical modeling* and *computer simulation* can help to understand the dynamics of the system under observation, its internal nature, and the possible future development. The effect of interactions on the system's environment can be tested [Klipp et al. 2009, Sec. 1.1]. The approach to understand biological systems combines both experimental and computational methods to answer questions of biological interest, focusing on the systems' behavior. *Computational systems biology* (CSB) applies computational methods on systems biology problems (i. e. it investigates systems biology by computational means). It focuses on the observation of properties such as a system's robustness and evolvability from the study of a biological network [Nature 2005]. Two important research tasks are:

1. the development of computational models that represent the biological world, and

[1]http://www.erasysbio.net/, last accessed 14 March 2011.

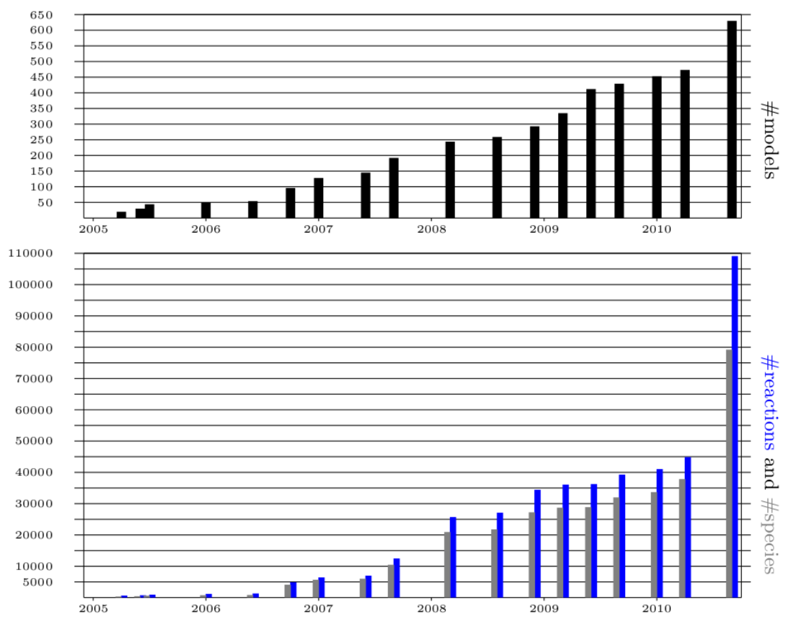


Figure 1.2.: Growth of BioModels Database during 18 releases, as of September 2010: The upper panel shows the number of models stored in BioModels Database at each release; the lower panel shows the number of distinct species and reactions encoded in the total number of models at each release. Adjusted from [Henkel et al. 2010].

2. the simulation and analysis of those models.

The study of a biological system can be represented by a *computational model* which can then be parametrized and simulated (Figure 1.1, blue box). *Computational models of biological systems* – especially the ones dedicated to the understanding of highly non-linear biochemical systems – proved to be very helpful (e. g. to map out promising experiments in the wet laboratory by analyzing simulation runs) [Köhn et al. 2009]. They 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 (e. g. few proteins that function as an amplifier) [Ferrell 2009].

The *number* of developed computational biology models grows quickly. BioModels Database, a repository for computational biology models, reflects this tendency (Figure 1.2, upper panel). The number of stored, curated models has increased from 22 in the year 2005 when BioModels Database was launched to 630 models in September 2010[2]. However, BioModels Database does not only grow in terms of the amount of

[2]`http://www.ebi.ac.uk/biomodels-main/static-pages.do?page=release_20100930`, last accessed 10 March 2011.

models, but also regarding the variety of modeled entities (Figure 1.2, lower panel). Furthermore, the modeled reaction networks increase in size, indicated by a raising average number of reactions per model[3]. The history of cell cycle modeling, for example, shows the increasing complexity of models for a given biological system: An XML-encoded version of the first "Tyson model" is available from BioModels Database[4]. It represents the published model in SBML format using nine species. One of its successors, the 1993 Novak/Tyson model [Novak and Tyson 1993] already encodes 14 species and 23 reactions[5]. Both models are single-compartmental. Since then more models on the cell cycle have been published; one example is the 2004 cell cycle model by Chen et al. [2004a]; its computational encoding is build of 74 species and 94 reactions[6]; it is still a single-compartmental model. The most complex model in BioModels Database in terms of number of encoded species and reactions as well as size of the model file is a model for the yeast molecular interaction network[7]. It contains 36263 species and 30965 reactions in one compartment; the XML file has a size of 50 MB.

1.2. Problem definition

The need to maintain computational biology models led to the incorporation of database and information systems techniques in dIEM oSiRiS (Figure 1.1, orange box). Information sharing is fundamental for the following investigations as it facilitates science [Piwowar et al. 2007]. Sharing of materials, methods, and data in the life sciences has received increased attention in recent years [Nordlie et al. 2009]. A research community that shares its data avails itself to viewing the data from different perspectives. Additionally, data sharing helps to identify errors in the data, discourages fraud and can be used in the training of new researchers [Piwowar et al. 2007].

Developing models is a time-consuming process; sometimes it can even be lifelong. A model on the function of a single protein can result in a PhD-thesis. Some researchers dedicate their career to "just one" model. For example, Guyton started developing a model on the "Relative importance of venous and arterial resistance in

[3]April 2005: 31.55 reactions/model; April 2010: 94.9 reactions/model (provided by Nicolas Le Novère)

[4]`http://www.ebi.ac.uk/biomodels-main/BIOMD0000000005`, last accessed 14 March 2011.

[5]`http://www.ebi.ac.uk/biomodels-main/BIOMD0000000107`, last accessed 14 March 2011.

[6]`http://www.ebi.ac.uk/biomodels-main/BIOMD0000000056`, last accessed 14 March 2011.

[7]`http://www.ebi.ac.uk/biomodels-main/MODEL3883569319` (DOI `doi:10.1371/journal.pone/0010662`), last accessed 17 February 2011.

controlling venous return and cardiac output" in the 1950s [Guyton et al. 1959]; he continued working on that model until his retirement in the 1980s [Montani et al. 1989]. The great effort put into model development is one reason to reuse a model also in other biological contexts. In order to test a new hypothesis *in silico*, it is desirable to take an existing model dealing with a similar or related system as a starting point. Considering previously developed models and components timely and qualitatively accelerates the model creation process (e. g. reoccurring structures and motifs such as feedback-loops [Wolf and Arkin 2003]). Also, studying existing models of a biological system helps to get an understanding of it. The fact that the number and complexity of models stored in public databases is increasing faster than the number of species and reactions encoded in the models suggests that the *duplication* of computationally encoded species and reactions is increasing, which is another justification for investigations in model reuse[8].

The need for model- and model component reuse is not doubted. Since 2006, model reuse has been repeatedly discussed as one major issue in CSB [Sauro et al. 2006; Le Novère 2006; Nordlie et al. 2009]. Nordlie et al. [2009], for example, emphasize the urge for reuse of models of neuronal networks, stating that "after more than 50 years of neuronal network simulations, we still lack [..] established practices for describing network models in publications. This hinders their [the models'] reuse".

Model reuse is already practiced, but it is limited. Famous examples for "phylogenies of models" include models on MAPK, Glycolysis, or the aforementioned cell cycle. In these examples, existing models were reused and then evolved into more complex representations of the system. However, only the knowledge about a model's *existence* eased its reuse and extension. Consequently, "quantitative models will be only as useful as their access and reuse is easy for all scientists" [Le Novère 2006]. I argue that modelers in the future need support in accessing models and following on their developments. When looking for existing models, it is essential to *find* them. Hence sophisticated model retrieval techniques are a prerequisite to model reuse (Figure 1.1, orange box).

The mentioned model retrieval relates to the well-known problem of information retrieval (IR). IR deals with techniques to efficiently find relevant data [Baeza-Yates and Ribeiro-Neto 1999]. Ranking then sorts the results according to the user's demands. A number of challenges arise when applying IR techniques on the retrieval of computational biology models. Some are relevant to all IR methods, for example, the identification of relevant data, the definition of a suitable format to store the

[8]Duplicated species include, for example, the chemicals C (Carbon), O_2 (Oxygen), H_2 (Dihydrogen), N (Nitrogen), or Fe (Iron). Examples for duplicated reactions include *oxidation* or *deduction*. Examples provided by Ron Henkel.

data that has been found relevant, or the development of suitable mechanisms to find the stored data and present it to the user in a meaningful manner. Others owe to the complexity of the biological knowledge encoded in those models, for example, the variety of data and scales or the incompleteness and uncertainty of the given data.

A model's meta-information is independent of the model encoding format and abstracts from the underlying modeling approach. Similar structures are used across different communities, mainly building on RDF and ontologies. This fact suggests to shift the model retrieval problem from the model-encoding XML level to the meta-information level. Once an extended storage concept for model meta-information exists the similarity determination can rely on the meta-information, instead of being limited to the information gained from the XML structure.

While model reuse is a recognized problem, the distinction between *reuse* and *retrieval* is often not sufficient: Finding a relevant model and *reusing* it are two different issues. Often, reusing a model implies studying it in the different parametrizations provided in the reference publication. Also, the information *how* to simulate the model (e. g. which simulation algorithm to use or what tasks to perform on the model) is necessary to reuse the retrieved model. A standardized format for the encoding of simulation information is therefore needed to describe the modifications on the model as well as the simulation setup.

1.3. Contribution

I have focused my work on *enhancing model reuse through better model retrieval*, thereby concentrating on two major aspects:

1. the *identification and appropriate storage* of standardized information and meta-information for the search process, and

2. the application of existing information retrieval techniques on *model retrieval*, incorporating the previously identified information.

My first contribution is a proposal for an *enhanced format to encode model meta-information* in XML-based model representation formats [Waltemath et al. 2011d]. The existence of such standard allows to encode meta-information on a model. Applying the standard to different model encoding formats allows for a comparison of models based on their meta-information.

Secondly, I identified experiment-related meta-information as one key aspect to model retrieval. I developed a *storage and exchange format for simulation experiments* applied on CSB models [Köhn and Le Novère 2008; Waltemath et al. 2011b]. Such a format enables the unified description of performed experiments, but also the experiment's exchange and reproduction. The information can be extracted and used for model retrieval. The basis of the developed XML language is a Minimum Information Guideline for simulation experiment descriptions which I developed in cooperation with a number of other institutes working in CSB [Waltemath et al. 2011a].

Thirdly, I identified further *relevant data and meta-information* for the characterization of computational biology models, and I defined the relational mapping on a database schema. A wide range of different data and meta-information on a model can thereby be stored, including the model structure, model annotations, but also model-related, experiment-related and model meta-information [Köhn 2009; Köhn et al. 2009; Waltemath et al. 2011c]. Further controlled vocabularies and ontologies were needed to encode parts of the identified information. My contribution is the development of a simulation algorithm ontology [Courtot et al. 2011].

My fourth contribution is the application of *information retrieval techniques* on "bio-model retrieval" [Köhn 2009; Köhn et al. 2009]. Using existing IR techniques, we developed a retrieval and ranking system that enhances the search for computational models of biochemical reaction networks [Henkel et al. 2010, 2011]. Retrieval is performed with focus on the meta-information of the models in question, not on the model structure alone [Köhn and Strömbäck 2008; Köhn et al. 2009]. The overall concept is used by the search system of the standard model repositories, BioModels Database [Henkel et al. 2010]. In order to evaluate the approach, I propose a framework called *Sombi* for the testing of such functions on different data bases [Waltemath et al. 2011c].

Additional problems, such as the evaluation of annotations, the evaluation of different ranking functions and ranking models for the use case of bio-model retrieval and so on are not part of this thesis, but will be investigated in future works.

1.4. Outline

I start by providing background information to the addressed problems (Chapter 2), and continue by elaborating on the state-of-the-art of model encoding, model annotation, model retrieval, and model storage (Chapter 3).

Chapter 4 describes an effort to enhance the current annotation scheme of SBML, a model representation format, by *standardizing the SBML-relevant meta-information* on a more detailed level. The result of this work is a proposal for an SBML extension package, called *annot package*.

Chapter 5 describes an approach to the formal encoding of simulation experiments as one important type of model-related meta-information (referred to as experiment-related meta-information). I introduce the set of guidelines identifying the Minimum Information About a Simulation Experiment (MIASE). A concrete implementation of those guidelines is described in full detail, and a storage solution for the demanded information is outlined.

Chapter 6 concentrates on the storage of model meta-information. It analyzes the requirements and introduces an extended storage concept for computational biology models. Experiences gained in the Annotation package development, but also during the study of different model representation formats led to the development of a database for *format independent, meta-information based bio-model storage*.

Chapter 7 shows how the information stored in *mDB* can be used to enable *format-independent ranked retrieval of bio-models*. The result of this work is a detailed retrieval concept with two existing implementations. The first one is a Lucene-based prototype implementation for *mDB*[9], and the second one a retrieval solution for BioModels Database[10]. Furthermore, I investigated a framework for the testing of different ranking functions on different model repositories, called *Sombi* .

Finally, Chapter 8 concludes and points to future prospects.

[9]Implementation by Ron Henkel, Diploma Thesis [Henkel 2009] (main supervisor)
[10]Implementation by Ron Henkel, Leonardo stipend, [Henkel et al. 2010]

2. Background

What is the difference between a live cat and a dead one? One scientific answer is 'systems biology'. A dead cat is a collection of its component parts. A live cat is the emergent behavior of the system incorporating those parts.

(Nature [2005])

This work is concerned with the enhanced storage and retrieval of computational biology models. Section 2.1 gives an overview of modeling basics in computational systems biology. A "model" can be considered a formal description of a system. Model representation formats enable exchange and reuse of those models; the two widely-known formats SBML and CellML, and the preliminary format πML are introduced in Section 2.2.

One way to provide further knowledge about the encoded system is to use model meta-information. The notion of meta-information is introduced in Section 2.3, including the concepts of controlled vocabulary, ontologies and the Resource Description Framework.

The technical background of this work are existing XML storage approaches (Section 2.4) and Information Retrieval techniques (Section 2.5).

2.1. Modeling and simulation in computational biology

Modeling and Simulation (M&S) is a traditional research field in computer science. A variety of definitions for the concept of a *model* co-exist; some stem from the 60s or 70s [Minsky 1965; Gottschalk et al. 1971; Eykhoff 1974], others are more recent [Fishwick 1995]. A generic definition is provided in [Cellier 1991, p. 5] (citing [Minsky 1965]):

Definition 2.1.1 (Model [Cellier 1991])**.** *A* model *of a system* S *and an experiment* E *is anything to which* E *can be applied in order to answer questions about* S.

Following [Cellier 1991, p. 5], the scope of the term model is here restricted to models that are codable as computer programs.

The above definition necessitates the further explanation of an experiment:

Definition 2.1.2 (Experiment [Cellier 1991]). *An* experiment *is the process of extracting data from a system by exerting it through its inputs.*

In order to apply the experiment to a model, we will consider the concept of simulation:

Definition 2.1.3 (Simulation [Cellier 1991]). *A* simulation *is an experiment performed on a model.*

The generic and concise definition of simulation describes it as a task that is applied to some entity. Here, the entity is a model, and the task is an experiment. [Cellier 1991, p. 4] explains experiments on a system as “a set of external conditions applied to the accessible inputs, and the observation of the reaction of the system to these inputs by observing the [..] behavior”. The model itself is a representation of a specific system under question on which such an experiment is performed.

Modeling lies at the heart of systems biology [Finkelstein et al. 2004]. A model in CSB summarizes established knowledge about a biological system in a coherent mathematical formulation [Klipp et al. 2007a, p. 5]. It furthermore explains the mechanism behind an observed behavior. A “model” is here regarded a *computational*, abstract representation of objects or processes of biological nature.

Definition 2.1.4 (Computational biology model (bio-model)). *A computational biology model is a formal description of the interactions in a biological system, mostly describing quantitative dynamics that allow for the modeled system to be simulated over time.*

For the scope of this work, the terms *model* and *bio-model* will be used interchangeably.

The modeling and simulation of biological systems supplements experiments in the wet laboratory; a suitable model can make useful and testable predictions [Klipp et al. 2009, Sec. 1.2]. The modeling process forces the researcher to describe his model's characteristics explicitly and in a formal manner. Modeling often reveals unspecified components or interactions, and it allows to stretch and compress space and time at will. Simulating a developed model for the aim of testing a hypothesis is cheap compared to biology experiments. Moreover, it is possible to simulate a system in a way that could experimentally not be done; a biological study is often based on a snapshot of the system while a simulation allows for the observation of a system's dynamics over time. The basic characteristics of a bio-model include [Klipp et al. 2009, pp. 6-9]:

- **Model scope**: A bio-model only considers certain aspects of the system, and disregards others. The scope of the model is within the considered aspects of the system, which should be chosen in a way that the disregarded properties do not compromise the basic answers obtained with the model.
- **System state**: The system state is a snapshot of the system at a given time and described by the set of variable values.
- **Variables, parameters, constants**: The variables, parameters and constants describe the used quantities and constitute the "model parametrization".
- **Model behavior**: The model behavior is determined by environmental influences and processes taking place within the system.
- **Model classification**: Processes can be classified with respect to a set of criteria, including quantitative, deterministic, discrete and others.

Kitano [2002a] defines further key properties for the understanding of a biological system at system-level: *System structures* representing the network of interactions, *system dynamics* describing how the system behaves over time under given conditions, *control methods* controlling the state of the cell, and *the design methods* used to modify and construct biological systems. Progress on the different aspects relies on a better understanding of computational sciences, including the better integration of such discoveries with existing knowledge [Kitano 2002a] (Section 3.1.2).

A bio-model is build of different parts, or components. While the fact that nature is modular on different levels is well-accepted, the identification of the specific biological constituents, and how they are identified is often not so clear. Fine-grained distinctions of model components have been proposed in literature, for example by Wolf and Arkin [2003] who distinguish several kinds of model parts (i. e. motifs, modules and games). As another example, Cooling et al. [2010] mention the importance of defining and characterizing standard mathematical model components (as opposed to biological components) for *in silico* systems definition, and proposes an online repository for Standard Virtual Biological Parts (SVPs) as part of the *CellML Model Repository* (Section 3.3.1). For the scope of this work, however, the general term *constituent* will be used for any part of a bio-model.

Definition 2.1.5 (Bio-model constituent, following [Le Novère et al. 2005])**.** *A* bio-model constituent *(constituent) is any identifiable part of the model, but also the model itself.*

The long tradition of M&S in Computer Science led to the development of a wide range of different *modeling approaches*. Some of them have also been applied on CSB problems. Besides kinetic models simulated and analysed, for example, with differential exuations, other computational approaches include process calculi, logical modeling, and Petri Nets [Priami and Quaglia 2004; Materi and Wishart 2007; Fisher and Henzinger 2007; Ewald et al. 2007].

One particular way to model concurrent processes in biological systems is the use of process algebras. The following paragraph introduces the π *Calculus* in more detail[1]. The π calculus, as process algebras in general, focuses on the continuation and communication of concurrent processes. A calculus is build of a number of defined processes and interactions, or rules, between those processes. Large systems can be modeled step-wise, making use of concurrent or parallel processes. Processes are named and connected via *channels*. A channel is a communication connection; one process acts as a sender, the other as a receiver. The message transferred over a channel is also named and therefore can be re-used in further interactions – allowing for the modeling of networks.

The following definitions are taken from [Milner 1999]. Channels are denoted with small letters $a, b, \ldots, x, y, z, \ldots$. *Local* channels are defined in one process and can only be used in the process they belong to. *Global* channels are valid for the whole π model and visible to all processes. Actions are denoted with capital letters $P, Q, R \ldots$. A number of actions form a process.

Definition 2.1.6 (π Calculus actions)**.**

$$\pi ::= x!(y) \tag{2.1}$$
$$\mid \quad x?(z) \tag{2.2}$$
$$\mid \quad \tau \tag{2.3}$$
$$\mid \quad [x = y]\pi \tag{2.4}$$
$$\mid \quad [x \neq y]\pi \tag{2.5}$$

$x!(y)$ denotes that y is the object that is sent via channel x (2.1). $x?(z)$ denotes that the object z is received via channel x (2.2). τ denotes that the action is not visible to other processes (no send or receive action) 2.3). $[x = y]\pi$ denotes that action π is executed if the condition $[x = y]$ is fulfilled (2.4). $[x \neq y]\pi$ denotes that the action π is executed if $[x \neq y]$ is fulfilled (2.5).

[1]The thorough description of the π Calculus does not reflect a specific role in the modeling of biological systems; it is necessary for the conceptual work on an XML-based standard format.

Definition 2.1.7 (π calculus syntax).

$$\begin{aligned} P ::= {} & 0 && (2.6)\\ \mid {} & \pi.P && (2.7)\\ \mid {} & P + Q && (2.8)\\ \mid {} & P \mid Q && (2.9)\\ \mid {} & (\nu x)P && (2.10)\\ \mid {} & !P && (2.11) \end{aligned}$$

0 denotes the finishing process (2.6). In $\pi.P$, P denotes a process. P can only be executed once π has finished. For example, in $x!(y).P$ P is executed once y has been sent over x (2.7). A full description is given in [Kühn 2008] (citing [Sangiorgi and Walker 2001]). $P + Q$ denotes a summation of two processes precluding each other (2.8). Either P or Q are executed. $P \mid Q$ denotes the parallel execution of two processes (2.9). Both, P and Q are executed. (νx)P denotes a newly created process (2.10). x is a new, unambiguous name in P. $!P$ denotes the infinite replication of process P.

Example 2.1.1 GenA = t?().(ProteinB | GenA) + a?().t?().GenA

Example 2.1.1 shows the definition of a process *GenA*. a and t are global channels. *ProteinB* is a process. The process definition of *GenA* implies that one of the two alternative executions (+) will take place when *GenA* is called:

- if any message is received on channel t (`t?()`) then the two processes *ProteinB* and *GenA* are executed in parallel (`ProteinB|GenA`).
- if any message is received on channel a (`a?()`) then the system waits until a message is received on t (`t?()`). Afterwards, process *GenA* (`GenA`) is executed.

The second option simply implements a delay. The first option calls *GenA* itself, but also a second process, *ProteinB*. A full example for the modeling of Euglena movement in dependency of light is given in Listing A.2 in Appendix A.2.2.

In an extension, exponentially distributed stochastic delays have been assigned to the interactions, which constituted the stochastic π Calculus [Priami 1995]. Further extensions developed within the graduate research school dIEM oSiRiS with regard to modeling problems in CSB include the *space* π [John et al. 2008a] or the *attributed* π [John et al. 2008b].

2.2. Representation formats

The application of mathematical concepts and M&S techniques to Biology demands techniques to store, integrate, and exchange models [Kitano 2002b; Le Novère 2006; Gennari et al. 2008]. *Standardization* plays a central role in facilitating the exchange and interpretation of the outcomes of scientific research, and in particular of computational modeling [Klipp et al. 2007b]; it is crucial to enable data, information and knowledge exchange. The definition of standards and guidelines maximizes a model's output and its scientific production. Several standards for systems biology have been proposed over the last years [Klipp et al. 2007b, suppl. results]. One such development is devoted to standardized representation formats for *bio-models* [Le Novère 2006]. Already in 1969, Garfinkel advocated

> establish[ing] a standard form of what a model should be like, how it should be described and documented [...]. This is intended in part to facilitate communication of information about models, which may be difficult owing to their complexity. [Garfinkel 1969]

A standardized, machine-readable format facilitates model exchange between users, databases and different simulation tools. Exchange formats enable the unified description of models, in the optimal case disregarding their particular underlying modeling approach. However, given the diversity of research questions and the number of different modeling frameworks used in CSB, there will not be a single standard description language that covers every aspect of biological modeling. Contrarily, rather specific standards with particular purposes will be used in the future [Strömbäck et al. 2007].

In this work, a *bio-model representation format* is considered any agreed-upon encoding of a bio-model in an exchangeable format. Agreement includes the code language, and the way the model is encoded in that language. Representation formats are typically defined using either the *Extensible Markup Language* (XML, [Bray et al. 2008]), or the *Web Ontology Language* (OWL, [McGuinness and van Harmelen 2004]). In the context of biological modeling, the most prominent representation formats are the standardized *Systems Biology Markup Language* (SBML) [Hucka et al. 2010] for the encoding of biological processes and the *CellML* for the encoding of cell biological processes. Further attempts for standardized representation of models include the *NeuroML* [Gleeson et al. 2010] for the encoding of neurophysiology models, but also formats for discrete event based models such as the *Biology Petri Net Markup Language* (BioPNML, [Chen et al. 2002]) for the representation of Petri Net models representing biological systems, or the π *Calculus Markup Language*

(πML, [Köhn and John 2007]) for the representation of π Calculus based models (Section 2.2.3).

2.2.1. Systems Biology Markup Language

The *Systems Biology Markup Language* (SBML, [Hucka et al. 2010]) 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. SBML has been adopted by more than 200 software systems from simulators to modeling tools and databases. As such it has been 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 [Hucka et al. 2010]. Minor revisions lead to a new SBML version, and each errata results to a new revision. The current version is *SBML level 3, version 1 core*. An SBML model[2] incorporates the following main parts [Hucka et al. 2010]:

- **Function definition**: A named mathematical function available for use in the model.
- **Unit definition**: A named new unit of measurement.
- **Compartment**: 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).
- **InitialAssignment**: 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**: Statement with an associated rate expression and describing a process that might change the amount of one or more species.

[2]In the following, an "SBML model" is considered being an SBML level 2, version 1 model.

- **Event**: Statement describing instantaneous, discontinuous change in one or more variables when a condition is triggered.

From level 3 on, SBML is modular, with a core definition and several proposed extensions (e. g. for multi-components or spatial information[3]). Sample SBML models are provided in Appendix A.2.

2.2.2. CellML

CellML [Cuellar et al. 2009] is an “implementation-independent model description language for specifying and exchanging biological processes” [Wimalaratne et al. 2009b]. Its language definition is available from the CellML Document Type Definition (DTD) [Cuellar et al. 2006]. The current version of the language is *CellML 1.1*, the successor of CellML version 1.0. CellML[4] supports a modular structure, allowing modelers to reuse parts of existing models [Cuellar et al. 2006].

CellML focuses on the mathematical formulation of biological processes. The main characteristics of a CellML model are the “explicit representation of modularity” and the “flexibility of the language” [Lloyd et al. 2004]. Both allow for the description of a diverse range of cellular and sub-cellular systems, including biochemistry, electrophysiology, system physiology and the mechanics of the intracellular environment [Lloyd et al. 2004].

CellML, in contrast to SBML, puts its major focus on the definition of components (i. e. functional units representing biological entities such as physical compartments or species, but also modeling abstractions [Cuellar et al. 2006]). From its resources[5], CellML now offers a set of standard components in CellML 1.1 format that through their public interfaces can be recombined into greater models of biological systems [Cooling et al. 2010]. A CellML model is build as a network of connections between self-contained components [Cuellar et al. 2006]:

- `<model>` **element**: The CellML root element, contains all the following elements

- `<component>` **element**: Smallest functional unit of a model, contains the variables and mathematics to describe the behavior of the subsystem

- `<connection>` **element**: Connects components to each other, and maps variables in one component to variables in another

[3]Further information is available from `http://sbml.org`, last accessed 14 March 2011.
[4]In the following, a “CellML model” is considered being a CellML 1.1 model.
[5]`http://models.cellml.org`, last accessed 14 March 2011.

- `<import>` **element**: Allows for import of further valid CellML models
- `<unit>` **element**: Allows for the definition of units, apart from standard units already provided; every variable and number has to have a unit assigned
- `<group>` **element** : Allows to define 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)

2.2.3. πML

Preliminary work on an exchange format for the π Calculus, called πML [Köhn and John 2007], shows how the encoding of π Calculus models is feasible. We propose an XML Schema that can be used to describe π Calculus models in a common XML-based encoding. It is available online from the sourceforge project[6]. The major building block of a basic πML-encoded bio-model are:

- channel definitions (global and private)
- nested process definitions
- initial process definitions

The language is build in a modular way, consisting of a core πML XML schema plus extensions for existing π Calculus variants. The current language has three extensions:

times The times extension allows to call a process n times, while the original πML schema needs to call each process separately, even the ones of the same type.

polyadic π The polyadic π Calculus allows to communicate more than one name over a channel. This πML module extends the original *definition* and *call* by additional lists of parameters. A number of parameters for that process can be defined with each new process definition.

stochastic π The stochastic π Calculus allows to assign rates to communications (i. e. assign the probability of that action to take place). The extension defines a `rate` attribute for local channels.

Listing 2.1 shows the simplest extension for the core πML language, the `times` extension. It redefines the original schema by extending the `callType` definition with the additional XML attribute `times`.

[6]`http://www.sourceforge.net/projects/piml`, last accessed 14 March 2011.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:redefine schemaLocation="PIML.xsd">
  <xs:complexType name="calltype">
   <xs:complexContent>
    <xs:extension base="calltype">
     <xs:attribute name="times" type="xs:positiveInteger"
      use="optional"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:redefine>
</xs:schema>
```

Listing 2.1: The *times* extension of the core πML language.

In [Köhn and John 2007] we presented a simple π Calculus model that simulates Euglena movements. Appendix A.2.2 shows the XML example complying with the XML schema for the basic π Calculus.

2.2.4. Further investigations

A range of further formats and format proposals exist. For example, computational neuroscientists investigate means for successful exchange of models and simulations [Nordlie et al. 2009]. The community is well-aware the benefits of sharing and model descriptions, and also simulation code, through model repositories and standard languages. It just recently published an updated specification for the standard model description language *NeuroML* [Gleeson et al. 2010]. The language exists in three *Levels* with different focuses: Level 1 describes the neuronal morphology and meta-data assigned to a model (MorphML). Level 2 extends Level 1 to describe also electrical properties of the model (ChannelML). Level 3 then provides constructs to describe the neuronal network of a model (NetworkML). NeuroML is XML-based and uses a modular approach to defining XML Schemas for the different language levels. In contrast to SBML and CellML, it defines very specific, high-level model features [Gleeson et al. 2010].

BioPNML is another example for an XML-based format fostering the standardized representation of biology models in Petri Net modeling. The format definition focuses on the representation of metabolic networks. It covers information on the network, but also encodes so-called exchange data such as for the graphical representation of the network.

For further information on model representation formats, Strömbäck et al. [2007] lists investigations beyond the above-named.

2.3. Meta-information

Existing representation formats focus on encoding the model structure, including the mathematical knowledge, interactions between the modeled entities and network structures. In SBML and CellML, the initial parametrization is also part of the model itself. However, one prerequisite for further working with a model is the incorporation of *meta-information.* It provides additional knowledge about the modeled system and enhances the computer-processed understanding of what is encoded in the XML strings. Hence modelers neither have to worry about details of file formats, nor do they have to dissect a model to understand what it is about [Le Novère 2006]. Meta-information are, for example, incorporated in the vitalization of models [Wimalaratne et al. 2009b], the transformation of models from one format into another [Villéger et al. 2010], the integration of models [Krause et al. 2009] and model-related data [Lambrix et al. 2009], the reasoning about models [Kell and Mendes 2008], the search for models [Henkel et al. 2010], or the model's presentation to a user in an information resource [Li et al. 2010; Swainston 2010].

2.3.1. Data – information – knowledge – meta-information

For the exploration of a model's meaning, different sources and levels of sources can be used. The following definitions are given to help in distinguishing them in the remainder of this work.

Data simply exists and has itself no significance beyond its existence [Bellinger et al. 2004]. Following Rowley and Hartley [2008], data is considered unprocessed, discrete facts or observations; it lacks context and interpretation. Classical database and information systems refers to *data* as the values of the different attributes in the database relations.

Example 2.3.1 For example, the sequencing of a DNA results in a set of *data*, which is a collection of letters 'A', 'G', 'C', 'T', 'N'.

Meta-data then is the explanation of the data. In Information Systems science, it is used to "store derived properties of media useful" [Kashyap and Sheth 1996]. In a database, the table- and attribute names are considered meta-data. Furthermore, the description of the data (e. g. in a data dictionary) can be referred to as meta-data.

Example 2.3.2 A possible meta-data for the above sequence of letters is an explanation what those letters stand for: A – adenine, G – guanine, C – cytosine, T – thymine, and N – non-defined.

Different types of meta-data can be distinguished. Following Kashyap and Sheth [1996], the relevant sub-groups that further describe the content of the document are:

content independent meta-data captures information that is independent of the content of the document it is associated with, for example *modification date*, or *location of a document.*

content dependent meta-data depends on the content of the document is associated with.

- **direct content-based meta-data** is based directly on the contents of the document, for example *full-text indices based on the text of the document.*
- **content-descriptive meta-data** describes the content of a document without direct utilization of the documents' content, for example *textual annotations.*
  - **domain-specific meta-data** describes the meta-data in the manner specific to the application or subject domain of information, for example *domain-specific ontologies.*

While domain-specific meta-data is related to a particular application area, all other types of meta-data reflect the format and organization of the underlying data [Kashyap and Sheth 1996].

The process of interpreting data leads to *information* [Lehner 1999]:

Definition 2.3.1 (Information)**.** Information *is data that has been given meaning by way of relational connection. (Ackoff in [Bellinger et al. 2004])*

In other words, data becomes information after it has been processed such as that it is then relevant for a specific purpose or context, and therefore is meaningful, valuable, useful and relevant [Rowley and Hartley 2008]. However, Lehner [1999] warns that providing information is no commitment to the truth of that information.

In computer parlance, a relational database makes information from the data stored within it [Bellinger et al. 2004]. In general, the rule applies that the better the data the more trustful the information gained from it [Lehner 1999].

Example 2.3.3 The processing of the above letters (data) will lead to DNA sequence information. The information gained from a sequencing data set could, for example, be `ATG` or `AUG`. It is specific to the context of DNA/RNA analysis.

Using techniques that collect relevant information and process leads to the gain of *knowledge*:

Definition 2.3.2 (Knowledge)**.** Knowledge *is the capacity for effective action in a domain of human practice. [Denning 2001]*

In contrast to information, knowledge involves understanding [Rowley and Hartley 2008]; it can be referred to as "justified true belief" [Lehner 1999].

Example 2.3.4 Understanding the above sample sequences ATG and AUG will lead to the knowledge that both are start codons of the DNA and RNA respectively. The knowledge about ATG might include the fact that it encodes "amino acid methionine (Met) in eukaryotes".

Knowledge by itself does not allow for further knowledge inference. Cognitive and causal processes are part of the *understanding*, that is taking knowledge and synthesizing new knowledge from it, as exemplified by Ackoff in [Bellinger et al. 2004]:

> [..] elementary school children memorize, or amass knowledge of, the "times table". They can tell you that "2 x 2 = 4" because they have amassed that knowledge (it being included in the times table). But when asked what is "1267 x 300", they can not respond correctly because that entry is not in their times table.

Dealing with understanding is not within the scope of this thesis.

In database and information systems, the *meta-information* level is rarely considered. One can, however, refer to meta-information as additional information on both, the data and the meta-data.

The CSB environment often refers to *annotations* as the thing that encodes meta-information and meta-data. The term annotation is a very general one, often denoting an additional, explanatory note – a "note of explanation or comment added to a text or diagram"[7] or, more generally, to a document. For the scope of this work an annotation is considered additional information (i. e. meta-information) on a model or a model constituent, mostly encoded using references to entries of ontologies. It may be provided together with the model or externally.

2.3.2. The concept of Ontology

Addressing the issues of terminological and conceptual conflicts is the task of *ontological engineering* – a research field studying methods and methodologies for

[7]New Oxford American Dictionary, Second Edition, Oxford University Press Inc.

ontology development. Ontologies have become particularly popular with the upstreaming work on the *semantic web* – the "extension of the current Web in which information is given a well-defined meaning by annotating Web content with ontology terms" [Lambrix 2005]. Another application area of ontologies is the use for meta-information encoding in the context of biological data and models.

In 1993, Gruber [1993] defined an ontology as: "a specification of a conceptualization", saying that it is considered "a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents"[8]. The generic definition has been further precised by Smith [2010]:

Definition 2.3.3 (Ontology)**.** *Ontology is the science of what is, of the kinds and structures of objects, properties, events, processes and relations in every area of reality. For an information system, an ontology is a representation of some pre-existing domain of reality which:*

1. *reflects the properties of the objects within its domain in such a way that there obtains a systematic correlation between reality and the representation itself*
2. *is intelligible to a domain expert*
3. *is formalized in a way that allows it to support automatic information processing.*

Barry Smith' definition of Ontology restricts the concept to *real world entities.* There exist oppositional definitions that see Ontology as a broader concept (see for example the discussion on "The Problem of Abstract Entities" in [Carnap 1956, pp. 13-22]). However, for the scope of biology ontologies used in the context of this work, we will follow the above definition. "Reality" is extended by mathematical terms and modeling concepts to apply Definition 2.3.3 on the *Systems Biology Ontology* (Section 3.1.2).

Ontologies in computer science In computer science, the term "ontology" refers to an

> engineering artifact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words [Guarino 1998].

[8]A detailed explanation of that short definition can be found on `http://www-ksl.stanford.edu/kst/what-is-an-ontology.html`, last accessed 01 May 2010. An explanation of Gruber's interpretation of the term "conceptualization" can be found in [Guarino 1997].

Ontologies are used without its philosophical aspects and rather concentrate on the specification of concepts of a domain of interest in a computer- and human-readable way, respecting the concepts' shared understanding [Stevens et al. 2010]. Opposed to the Ontology concept introduced above, an ontology in computer science is language-dependent. Therefore, two ontologies can use different vocabulary but still share the same conceptualization.

Apart from specifying terms and relating them to each other, ontologies' benefits further include the "reuse, sharing and portability of knowledge across platforms, and improved maintainability, documentation, maintenance, and reliability" [Lambrix 2005].

Ontologies in modeling and simulation The use of ontologies for the linkage of knowledge and information in M&S has already been advocated [Miller et al. 2004; Fishwick and Miller 2004; Silver et al. 2007; Benjamin et al. 2006, 2007]. The advantages of using ontologies in M&S include the possibility to archive taxonomic properties, for educational purposes, but also to distinguish instances of a particular model [Fishwick and Miller 2004]. One important application of ontologies in M&S is the increase of interoperability, integration and reuse of simulation artifacts (i. e. libraries, components, simulators, simulation tools, and models) [Miller et al. 2004]. The current approaches to use ontologies mainly focus on the simulation model development task.

Ontologies in computational systems biology *Biology ontologies* (bio-ontologies) encode biological knowledge in a structured way. They mainly concentrate on: biological *entities* (e. g. the CheBI term `CHEBI:25699 organic ion`), on biological *functions* (e. g. the GO term `GO:0043167:ion binding`), and on biological *processes* (e. g. the GO term `GO:0051516:regulation of bipolar cell growth`). One advantage of ontologies is that a concept can be used to handle ambiguous terms and synonyms which particularly in Biology occur often. On the one hand, several *synonyms* might exist for one biological thing (e. g. "Glucose synthesis", "Glucose biosynthesis", "Glucose formation" or "Gluconegenisis"); on the other hand, the same name might refer to different biological things depending on the particular context (e. g. the phrase "bud initiation" which might be related to "flower bud initiation" in flower development, to "cell bud initiation" in cell development, or "tooth bud initiation" in tooth development)[9]. To give another example, there are on average seven synonyms for a single human gene. The one with most synonyms

[9]Example provided by Janna Hastings during the EBI Roadshow Workshop `http://www.ebi.ac.uk/training/roadshow/100924_rostock.html`, last accessed 14 March 2011.

counts 164 different terms[10]. The situation demands the definition of controlled terms to handle different and identical concepts. Particular bio-ontologies will be introduced in Section 3.1.2.

2.3.3. MIBBI

While representation formats (Section 2.2, Section 5.4) describe how to encode the information pertaining to the models, they do not cover what to encode. Defining rules for the latter is the role of so-called *Minimum Information* (MI) guidelines. The notion of MI guidelines had originally been introduced to Biology to "clearly describe an experiment and report the variables necessary for data analysis" [Quackenbush 2006].

The initiative behind MI checklists is the global *Minimum Information for Biological and Biomedical Investigations* project (`http://www.mibbi.org`). MIBBI is coordinated by representatives of different communities developing MI checklists, or guidelines. The web-based resource lists the existing MIs for the field of biological and biomedical investigations, provides access to them and aims at improving the transparency, accessibility, but also the interoperability of existing checklists which leads to a unification of the standardization community [Taylor et al. 2008].

A guideline for minimum information is not intended as a substitute for production protocols and procedures that are documented elsewhere; the provided information will usually not be sufficient for reproduction, but it will clearly identify the relevant entities of an experiment [Sherman 2009]. As such, the MI guidelines are a helpful mean for the semi-formal description of rules for the repetition of experiments (in the very broad sense) and other procedures particularly used in the fields of computational biology and biology.

While the sets of rules are human-understandable and descriptive, they cannot be used for automatic processing in a computational environment. Consequently, the guidelines need to be complemented by a certain data model in order to be "used". For MIRIAM, there exist several data models that comply with the MIRIAM guidelines. One example is SBML (Section 2.2).

2.3.4. Resource Description Framework

The *Resource Description Framework* (RDF, `http://www.w3.org/RDF/`) is "a language for representing information about resources" [Miller and Manola 2004], in particular for representing meta-data about Web resources in the World Wide Web. The RDF primer generalizes the concept of a "Web resource" to "information about

[10]`http://blog.novoseek.com`, last accessed 26 March 2010.

things that can be identified on the Web, even when they cannot be directly retrieved on the Web" [Miller and Manola 2004, Sec. 1]. RDF reuses ideas from knowledge management, artificial intelligence, data management, logic-based knowledge representation, relational databases and more [Miller and Manola 2004, Sec. 2.5]. One advantage of the standardized and formal encoding of knowledge through RDF statements is that it can be computationally processed by applications. The common framework provided by RDF to express the information in a standardized way leverages the exchange of information between different applications.

RDF statements The basic concept of RDF is the identification of *things* using *Web identifiers*, so-called *Uniform Resource Identifiers (URIs)*. The resources are described by *properties* with particular *property values* [Miller and Manola 2004]. The specific terminology used in RDF is (taken from [Miller and Manola 2004, Sec. 2.1]):

Definition 2.3.4 (subject)**.** *The part that identifies the thing the statement is about is called the* subject*.*

Definition 2.3.5 (predicate)**.** *The part that identifies the property or characteristic of the subject that the statement specifies is called the* predicate*.*

Definition 2.3.6 (object)**.** *The part that identifies the value of a property is called the* object*.*

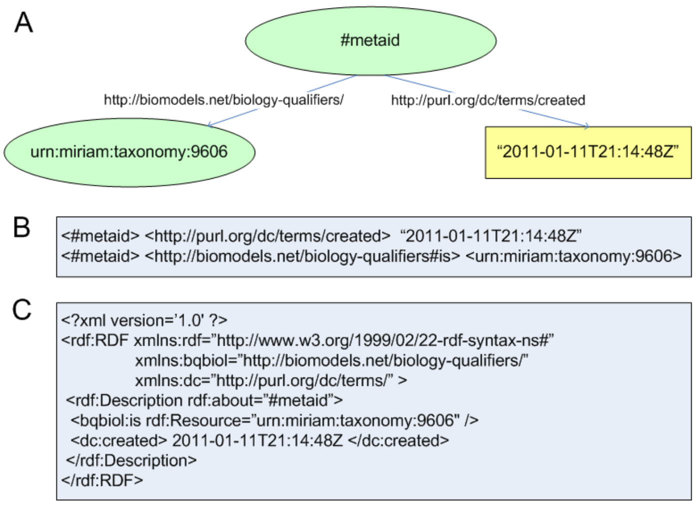
Because of the generality characteristic of URIs, they are used in RDF to identify subjects, predicates, and objects in statements. Miller and Manola [2004] extend the concept of URIs to *URI references* [Miller and Manola 2004, Sec. 2.1].

Definition 2.3.7 (URIref (URI reference), adapted)**.** *A URI reference (or URIref) is a URI, together with an optional fragment identifier at the end. The fragment is separated by the # character.*

A *resource* is then:

Definition 2.3.8 (resource)**.** *A resource is defined as anything that is identifiable by a URI reference).*

Objects in RDF may either be URIrefs, or constant values, so-called *literals*. Both, subject and predicate must not be literals, they always have to be URIrefs. Using URIrefs as subject, predicate and object in statements supports the development and use of shared vocabularies on the web [Miller and Manola 2004]. One advantage


A
#metaid
http://biomodels.net/biology-qualifiers/
http://purl.org/dc/terms/created
urn:miriam:taxonomy:9606
"2011-01-11T21:14:48Z"
B
C

```
<#metaid> <http://purl.org/dc/terms/created>  "2011-01-11T21:14:48Z"
<#metaid> <http://biomodels.net/biology-qualifiers#is> <urn:miriam:taxonomy:9606>
```

```
<?xml version='1.0' ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:bqbiol="http://biomodels.net/biology-qualifiers/"
            xmlns:dc="http://purl.org/dc/terms/" >
 <rdf:Description rdf:about="#metaid">
  <bqbiol:is rdf:Resource="urn:miriam:taxonomy:9606" />
  <dc:created> 2011-01-11T21:14:48Z </dc:created>
 </rdf:Description>
</rdf:RDF>
```

Figure 2.1.: The RDF representation as a graph with nodes and arcs representing the resources, and their properties and values (A). The triplet representation of the graph (B). The RDF representation in RDF Syntax RDF/XML for recording and exchanging the graphs (C).

of using URIrefs for statement definitions is that an URIref allows for the more precise identification of a thing than using a sole string. Another advantage is that a thing with a URIref assigned can be further described by other RDF statements, while a string chain (i. e. a literal) cannot.

The sample statement "`#metaid` is `http://purl.org/dc/terms/created` whose value is `2011-01-11T21:14:48Z`" defines the subject with a URIref. The predicate is represented by the URIref `http://purl.org/dc/terms/created`, and the object is a given by the literal `2011-01-11T21:14:48Z`.

RDF notations RDF allows to model the encoded information in different ways (Figure 2.1).

The *graph representation* uses nodes and arcs (Figure 2.1, A). An RDF graph is formed based on the idea that "the things being described have properties which have values" [Miller and Manola 2004, Sec. 2.1]. Resources are described by making statements about those properties and values: The nodes in an RDF graph represent the subject and object of a statement; the arc represents the predicate. Each arc is directed from subject node to object node. Ellipses in the RDF graph represent URIrefs (e. g. `#metaid`), while boxes represent literals (e. g. `2011-01-11T21:14:48Z`).

The *triplet notation* (Figure 2.1, B) offers an alternative to the graph representation, for example if a graph gets to inconvenient to be drawn. Each statement of the graph is written as a single triplet, consisting of the subject, object and predicate (in that order). A triple describes a single arc in the graph, with the subject being the arc's beginning and the object being the arc's ending. URIrefs are put in angle brackets (e. g. `<#metaid>`), while literals are put in quotes (e. g. `"2011-01-11T21:14:48Z"`).

Furthermore, RDF uses XML to represent statements in a machine-processable way (Figure 2.1, C). The syntax for writing RDF is called *RDF/XML* [Beckett 2004, Sec. 3]. The description of a statement is enclosed in an `rdf:RDF` XML element. The statement itself is enclosed in an `rdf:Description` element, being regarded a description about the subject of the statement. The subject is referred to in the `rdf:about` attribute inside the `rdf:Description` element (e. g. `rdf:about="#metaid"`). Nested within the containing `rdf:Description` element is the property element representing the predicate (e. g. `<dc:created />`) and object of the statement (e. g. `2011-01-11T21:14:48Z`). The nesting indicates the application of the property on the subject. More details on the RDF/XML syntax are given in [Beckett 2004].

RDF containers and collections Often, there is a need to describe a group of things in RDF. The framework provides four different concepts to encode grouped statements, including the three different *RDF containers* `rdf:Bag`, `rdf:Seq` and `rdf:Alt`, and the *RDF collection* `rdf:List` [Miller and Manola 2004, Sec. 4].

rdf:Bag A resource having the type `rdf:Bag` represents a group of resources or literals [..] where there is no significance in the order of the members.

rdf:Seq A resource having the type `rdf:Seq` represents a group of resources or literals [..] where the order of the members is significant.

rdf:Alt A resource having the type `rdf:Alt` represents a group of resources or literals that are alternatives (typically for a single value of a property).

rdf:List A resource having the type `rdf:List` represents a group of resources or literates that consists only of the specified members.

While bags, sequences and alternatives are open (i. e. there is no way of stating that they cover all members of a group) the list collection defines a closed group of resources.

2.4. Storage of XML documents

The focus for this work is on bio-models encoded in standardized XML formats. XML documents can be classified into three types [Klettke and Meyer 2003a]: data-centric, document-centric and semi-structured. *Data-centric* XML documents are regular and structured documents with typed data (e. g. XML-encoded gene sequence data) . Contrarily, *document-centric* documents contain predominantly full text, mostly designed for human readers (e. g. XML-encoded brochures or text books); they can be structured differently. Finally, *semi-structured* documents are a mixture of the two former types, containing characteristics of both, i. e. pure data but also textual descriptions [Klettke and Meyer 2003b]; the majority of bio-models considered in this work is semi-structured.

A work on XML documents necessitates methods for efficient storage and access of documents and document parts, the support of standardized XML query languages, standardized interfaces to applications such as DOM, and multi-user support, particularly transaction support, recovery and resynchronization [Helmer et al. 2003]. General storage concepts for XML documents include [Klettke and Meyer 2003a]:

- Storing the XML documents as data files with additional indexing of the document context (full text index) and structure (structure index)
- Storing the XML graph structure (simple or Document Object Model structure)
- Mapping the XML document structure on a relational or object oriented database

The approach of indexing the XML documents is often taken for documents with a document-centric structure. Here, native XML databases or DBMS supporting XML as a data type can be used. They keep the XML document in its original form. Furthermore, IR techniques can be applied and the document's mark up can be used for the interpretation of the XML document (e. g. `<species>` in an SBML document). A schema definition is not necessary to store the documents [Klettke and Meyer 2003a].

The mapping on a relational database schema is the preferred approach for data-centric documents; a number of automatic as well as user-defined mapping solutions exist. One prerequisite is the existence of an XML schema which is used to define the mapping on the database schema. The full documents are not kept. Consequently, a document re-creation is only partially realizable. The advantage of this approach lies within the availability of techniques for the automatic mapping of the XML document structure on a database structure. Furthermore, SQL queries are available for later search on the data [Klettke and Meyer 2003a].

Typically, semi-structured XML documents are stored as graph structures. Therefore, all XML elements and their values are stored in a relation, together with their predecessors. A second relation stores the XML attribute names and their corresponding values. The attributes are then linked to the elements they occur in. An advantage of this storage approach is that an XML schema definition is not necessary. However, the document re-creation is only hardly possible and demands a high effort. The approach only allows for XQuery-like queries on the data. SQL has to be adapted to work on the document tree. Updates and queries can be easily realized with existing DOM methods [Klettke and Meyer 2003a].

Hybrid and redundant XML storage The above-introduced storage approaches for XML documents can also be combined into hybrid approaches [Klettke and Meyer 2003a]. This is often necessary if XML documents have to meet the demands of more than one application. Also, if the document character cannot clearly be determined, hybrid approaches are an appropriate solution. For example, in SBML (Section 2.2.1), data-centric encoding of the list of species (`listOfSpecies`) can be mapped on a relational schema; the document-centric encoding of SBML notes (`notes`) is stored using an indexing approach.

Section 6.3 discusses the various storage options and their advantages and disadvantages for use with bio-models in full detail.

2.5. Retrieval

Chapter 1 has already mentioned the increasing complexity and number of computational models of biological systems (Figure 1.2 on page 5 shows the numbers for BioModels Database). Because of the size of systems under consideration, as well as their multi-scale aspects, modeling activity in integrative systems biology requires researchers to leverage new approaches from prior work [Waltemath et al. 2011a]. In addition, the modeled systems often build on each other or interact. Many groups work on similar biological questions, often in close cooperation. All these facts call for the reuse of existing models and their associated simulation experiments.

Reuse in general is considered "the action of using something again or more than once"[11]. More specifically, reuse in *Software Engineering* is referred to as "the isolation, selection, maintenance and utilization of existing software artifacts in the

[11]New Oxford American Dictionary, A. Stevenson and C.A.Lindberg (Ed.), Second Edition

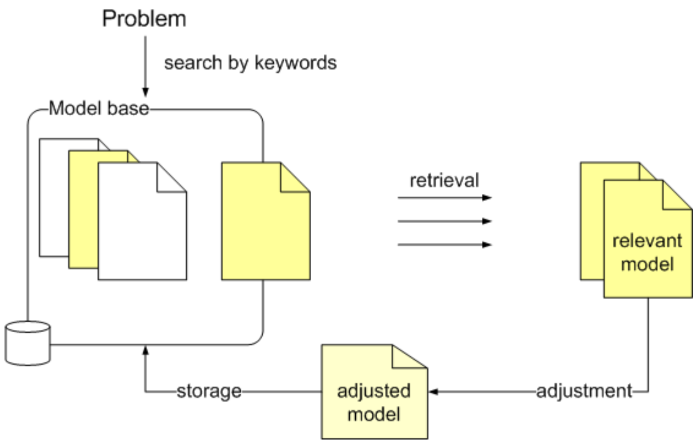


Figure 2.2.: The model management workflow, showing the role of search, retrieval and storage.

development of new systems" [Robinson et al. 2004] (citing [Reese and Wyatt 1987]). Brash [2001], in the context of *Information Systems development*, defines reuse as "the employment of a previously used and explicitly defined systems development artefact's in another information systems development process". Although the most common form of reuse is practiced on code level, reuse can be applied to different stages of development, including the requirements specification, the model design, its implementation and testing [Robinson et al. 2004] (citing [Reese and Wyatt 1987]).

A successful *retrieval* is the prerequisite for model reuse. Figure 2.2 shows how the retrieval problem is relevant for the systems biology workflow. Given a particular question, hypothesis or problem, the model base can be searched for related models dealing with that case. Ideally, a number of relevant models are retrieved and adjusted (i. e. reused in a similar context). More precisely, a particular problem is formulated. It represents the biological modeling question of interest. Keywords are determined and then used to query the *model base*. The result is a set of *retrieved models*. They are reused to solve the problem, for example by adaptation of the existing models, or by merging them. The newly developed model undergoes the typical verification, testings and potential refinements and corrections. Afterwards, it is stored either as a new version or a new model in the model base.

In this work, the problem of retrieving relevant information from a given data- and information base is partially addressed by using database techniques, but also grounded on *Information retrieval* (IR) techniques. IR is the process of recovering "an information stored in a system (i. e. a database) on users demand" [Ferber 2003]. Vannevar Bush's article "As we may think" [Bush 1945] first popularized the IR idea. Interestingly, Bush already then used an example from Biology to motivate the IR problem, saying that

> "Mendel's concept of the laws of genetics was lost to the world for a generation because his publication did not reach the few who were capable of grasping and extending it; and this sort of catastrophe is undoubtedly being repeated all about us, as truly significant attainments become lost in the mass of the inconsequential".[12]

IR supports vague queries on different kinds of documents, including text, image, or music documents. Queries can be semantically more complex than SQL statements; when searching a document collection, especially containing multimedia documents, the corresponding data bases are often searched by the document's characteristics.

While data retrieval queries typically return *exactly* matching data sets (i. e. a data set is checked for the existence or absence of a particular entity), IR-techniques also include *vague* or similar matches in the query result [Van Rijsbergen 1979]. A data retrieval system searches for data by exactly defined, *complete* queries, mostly using *artificially* constructed SQL statements (`SELECT FROM WHERE`). Opposed to that, IR systems allow to query the data vaguely without formally defining the query (Find all documents dealing with ...). The query in that sense is an *incomplete* query in *natural* language.

However, a certain error tolerance must always be accepted when using IR approaches, whereas in data retrieval all search results are relevant to the query [Ferber 2003]. An advantage of data retrieval methods over IR methods is that queries can be *explicitly* answered by accessing the values of the corresponding attribute(s). The data is stored in a structured way and can be directly queried. In an IR system, the data is stored in an unstructured way; the demanded information to query is often only *implicitly* given [Ferber 2003] and necessitates a preprocessing and analysis of the documents.

Finally, data retrieval queries result in a *set* of equally relevant data sets. On the contrary, IR queries result in a *list* of relevant documents, sorted by their relevance regarding the query.

Information retrieval models

Information retrieval models (IR model)[13] are existing IR based methods for the retrieval of documents. They are defined as [Baeza-Yates and Ribeiro-Neto 1999,

[12]Mendelian Inheritance, first published in 1865 and 1866, and then re-discovered in 1900 `http://en.wikipedia.orgh/wiki/Mendelian_inheritance`, last accessed 14 March 2011.

[13]The term "model" must not be confused by the definition for a bio-model given earlier.

p. 23]:

Definition 2.5.1 (Information retrieval model)**.** *An information retrieval model is a quadruple (D,Q,F,R(q_i, d_j)) where*

1. *D is a set of logical views/representations for the documents in the collection*
2. *Q is a set of logical views/representations for queries*
3. *F is a framework for modeling document representations, queries and their relationships*
4. *R(q_i, d_j) is a ranking function defining ordering among the documents d_j with regard to the query q_i*

For the IR models considered in this work, the representation of a document $d_i \in D$ consists of a set of *words*. Processed words are called *terms*. IR systems typically categorize the extracted terms into so-called *features*, that is "information extracted from an object" [Baeza-Yates and Ribeiro-Neto 1999, p. 442] where the information has some prominent attribute or property.

A query $q_i \in Q$ is considered a set of words which has been transformed into a set of terms. Transformations include typical preprocessing methods such as lexical analysis, stopword removal, stemming and others [Baeza-Yates and Ribeiro-Neto 1999, pp. 165-73].

IR systems can be classified by the underlying *IR models*. Kuropka [2004] provides an overview of existing IR models (Figure 2.3). He distinguishes IR models by (1) the mathematical model used, and by (2) the properties of the model. This categorization follows the suggestions of [Baeza-Yates and Ribeiro-Neto 1999]. The three different mathematical bases are:

- *Set-theoretic models* use set operators for the determination of similarities between a document and a query, e. g. $\cap$, $\cup$.
- In *algebraic models*, both query and document are represented as some algebraic mean such as vectors, matrices or tuples. The similarity is then calculated using scalar products, cosine measures and others.
- Finally, *probabilistic models* determine similarities between document and query as the probability for the relevance of a document for a given query.

Furthermore, [Kuropka 2004] distinguishes IR models by their ability to take term interdependencies into account:

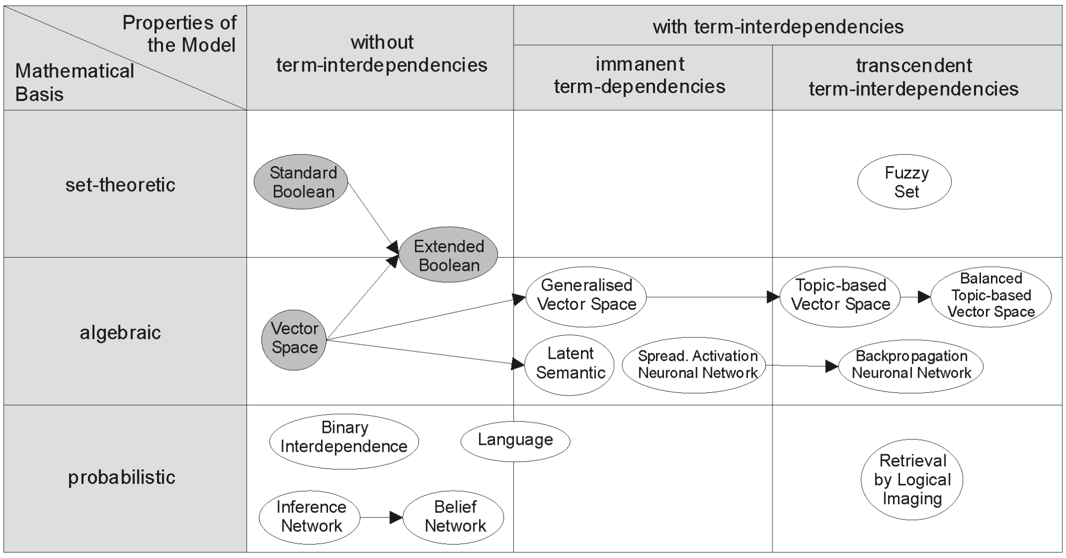


Figure 2.3.: Categorization of IR-models by Kuropka [2004]. IR-models used in this thesis are marked in gray.

- Models *without term-interdependencies* consider each term an independent entity.
- Models *with immanent term-interdependencies* define the interdependencies between the terms themselves.
- Models *with transcendent term-interdependencies* have term interdependencies defined by externals, e. g. the user.

From the classification of IR models given in [Kuropka 2004], we will here only consider the *Standard Boolean Model* (SBM) and the *Vector Space Model* (VSM).

Standard Boolean Model The standard IR model is the simple yet powerful *Standard Boolean Model* (SBM, [Baeza-Yates and Ribeiro-Neto 1999], pp. 25-27). Following the categorization by Kuropka [2004], the SBM is mathematically a set-theoretic model, without any term-interdependencies (i. e each term is considered an independent entity); it builds on Boolean algebra. The term weights are all binary (i. e. 0 or 1). A query q is a conventional Boolean expression in disjunctive normal form. A document is relevant if it fulfills the requirements of the query q. Otherwise, the document is predicted to be irrelevant.

It is defined as (definition 2.5.2):

Definition 2.5.2 (Standard Boolean Model, [Baeza-Yates and Ribeiro-Neto 1999],

p. 26). *For the Boolean model, the index term weight variables are all binary i. e.* $w_{i,j} \in \{0,1\}$. *A query* q *is a conventional Boolean expression. Let* $\vec{q}_{dnf}$ *be the disjunctive normal form for the query* q. *Further, let* $\vec{q}_{cc}$ *be any of the conjunctive components of* $\vec{q}_{dnf}$. *The similarity of a document* d_j *to the query* q *is defined as*

$$sim(d_j,q) = \begin{cases} 1 & \textit{if } \exists\ \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall k_i,\ g_i(\vec{d}_j) = g_i(\vec{q}_{cc})) \\ 0 & \textit{otherwise} \end{cases}$$

If $sim(d_j,1) = 1$ *then the Boolean model predicts that the document* d_j *is relevant to the query* q *(it might not be). Otherwise, the prediction is that the document is* not *relevant.*

The two main advantages of the SBM are its clearly defined IR model and its simplicity [Baeza-Yates and Ribeiro-Neto 1999]. A restriction of the SBM, however, is that it only distinguishes between "relevant" and "not relevant" documents. As such, a result *ranking* cannot be realized. Also, a *weighting* of single query terms is not possible, as for example in expressing that "it is particularly important that the author of the document is `Goldbeter` and only a minor requirement that the model describes the `cell cycle`". Queries on a system containing a large set of similar, domain-specific documents (e. g. a database for bio-models) might retrieve only a small subset of documents which still contains several hundreds of unsorted hits.

To avoid some of the disadvantages, the standard boolean model can be improved, for example, when using the *Extended Boolean Model* [Schmitt 2005]. An alternative is the use of other IR models.

Vector Space Model The Vector Space Model (VSM) was first described in 1975 by Salton et al. [1975]. It is mathematically an algebraic model, without term-interdependencies [Kuropka 2004]. Using the vector space model, each document d_i and query q_i are represented by a vector. Each dimension in the vector space corresponds to a term t_i. A formal definition of the Vector Space Model is given by [Schmitt 2005]:

Definition 2.5.3 (Vector Space Model [Schmitt 2005]). T *is a set of terms where* $T = \{t_1, ..., t_n\}$ *and* D *is a set of documents where* $D = \{d_1, ..., d_m\}$. *For each document* $d_i \in D$ *exists a term* $t_k \in T$ *with a weight* $w_{i,k} \in \Re$. *The weights of* d_i *can be combined in a vector* $w_i = (w_{i,1}, ..., w_{i,n}) \in \Re^n$. *The vector* w_i *describes a document in the vector space model it is called document vector. The* ***query vectors*** *are constructed in the same way. A query* q *is represented by* $q \in \Re^n$. *The similarity between a document and a query is computed using a similarity function* $s : \Re^n \times \Re^n \to \Re$.

The complexity of the VSM reveals particularly with large documents containing many terms. All documents have to be considered for a query and evaluated for relevance. Furthermore, queries of high terms dimensionality result in a high-dimensional vector space. Another disadvantage of the VSM is, as is true for the SBM, that it does not consider the document structure (e. g. the word order). The characteristic of term-independency regards all formed vectors as independent of each other, while terms are often related to each other ("signaling pathway").

For a limited set of documents with a reasonable number of terms, however, the VSM reliably ranks documents relative to a given query vector. The dimensions can be reduced by preprocessing methods (e. g. stopword removal) or by corpus reduction through preprocessing with the SBM.

Similarity measures

There exist different measures to determine the similarity between two documents [Ferber 2003], including the scalar product or the cosine similarity.

The cosine similarity determines the similarity of two vectors by measuring the cosine of the angle between them. The bigger that similarity value the better a match. A cosine of '0' denotes orthogonality of two vectors, thus indicating that the corresponding documents are not similar. The retrieved objects are sorted in descending order according to their $sim_{cos}(d, q)$ value. To measure the dissimilarity of objects, the *Euclidean distance* can be used. It calculates the distance between two documents in the vector space. A distance of '0' denotes perfect matches. Returned objects are sorted in ascending order according to their $dissim(d, q)$ values.

Multimedia retrieval

One example for the use of IR techniques is *multimedia retrieval.* Existing *Multimedia Information Retrieval models* (MIR models) describe songs, images, videos and other multimedia documents with different kinds of information, including meta-information like the document author, title, spectral information, or keywords [Zhang et al. 2009]. Common measures to determine the similarity between multimedia documents exist. *Metadata-based similarity measures* (MBSM) define queries by connecting keywords gained from the media object with Boolean operators like $\wedge, \vee$. They use text retrieval techniques to compare these query keywords with features of the model. *Content-based similarity measures* (CBSM) utilize so-called low-level features (or automatically extractable items) such as rhythm and incorporate them in the queries to search the content of music pieces. Different methods have been developed to retrieve the items represented by low-level features (e. g. humming, tapping or query-by-example). *Semantic-description-based similarity measures* (SDSM) evaluate meta-information on multimedia objects that are described with predefined words of different vocabularies.

2.5.1. The information retrieval process

The IR process may be split into (1) the *preparation process* which includes building the collection from existing documents and creating the indexes, and (2) the *retrieval and ranking process* which includes analyzing the queries, retrieving the relevant documents, and ranking them. Two sample documents are used for illustration:

Example 2.5.1 Document d_1 contains the sentence *"DKK1 antagonizes WNT signalling during head formation in mice."*[14]. Document d_2 contains the sentence *"In the mouse, Ror2 and Ror1 knockout phenotypes resemble those of Wnt5a/ null mice"*[15].

Collection creation from existing documents To support queries on unstructured or semi-structured documents, the original documents are preprocessed: First, each document is split into a set of words. These words are then transformed into terms, using well-developed standard methods such as stemming, stopword removal and alike. Structural information is lost during this process, as document elements are considered mostly detached from their context. We refer to the set of documents d_i with $\sum d_i = D$ as a *collection.*

Example 2.5.2 The result of the first preprocessing step for the sample documents d_1 and d_2 might, depending on the particular algorithm, lead to the following list of *words*:

$$\begin{aligned} d1 = {} & \{DKK1, antagonizes, WNT, signalling, during, head, formation, in, mice\} \\ d2 = {} & \{in, the, mouse, Ror2, and, Ror1, knockout, phenotypes, resemble, those, of, \\ & WNT, null, mice\} \end{aligned}$$

The list of words is then processed to contain the following list of *terms* for d_1 and d_2, respectively[16]:

[14]taken from [Klaus and Birchmeier 2008].

[15]taken from [Nusse 2008].

[16]This example uses the Porter Stemmer and stopword removal. The Porter stemmer is one of the most commonly used stemming algorithms; it realizes context sensitive suffix removal and has first been described in [Porter 1980].

$$\begin{aligned} d1 = &\ \{DKK, antagon, WNT, signal, dure, head, format, mice\} \\ d2 = &\ \{mous, ror, knockout, phenotyp, resembl, WNT, null, mice\} \end{aligned}$$

Index creation One or more *indices* are built from the collection representation (Example 2.5.2, bottom). The *inverted index* (or inverted list) is the most prominent indexing method [Cutting and Pedersen 1989]. It is a "text index composed of a vocabulary and a list of occurrences" [Baeza-Yates and Ribeiro-Neto 1999, p. 445]; each index entry contains the term derived from the words in the document and a list of associated documents[17]. Inverted indices can show different kinds of structure. For example, a standard inverted file index only indicates the document in which a word appears while a *full* inverted index also includes the exact position of the term within the document. This information enhances the retrieval process. Further information can be added to the $(term, document)$ tuples, including the term frequency, or the inverse document frequency. While the index size is increased, such additional information allows for a more sophisticated retrieval process, as it can include, for example, phrase search, if the order of terms is stored.

For each word occurring in a document that documents' ID is stored in the corresponding entry in the list of terms. The following example shows the inverted index for d_1 and d_2.

Example 2.5.3 The inverted index with terms t_i for the above documents d_1 and d_2 is (extract):

$$\begin{aligned} t_1 = &\ antagon \,|\, d_1 \\ t_2 = &\ DKK \,|\, d_1 \\ t_2 = &\ dure \,|\, d_1 \\ t_3 = &\ mice \,|\, d_1, d_2 \\ t_3 = &\ mous \,|\, d_2 \\ t_4 = &\ phenotyp \,|\, d_2 \\ t_5 = &\ WNT \,|\, d_1, d_2 \end{aligned}$$

and so on.

[17]NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, `http://www.itl.nist.gov/div897/sqg/dads/HTML/invertedIndex.html`, last accessed 30 June 2010

Query analysis The two processes of *building the collection* from existing documents (i. e. extracting words from the document and processing them into terms for storage in the index) and *processing user queries* (i. e. converting them into terms for the query of the index) must rely on the same algorithm. That is why, the preprocessing of query words is directly dependent on the chosen preprocessing applied for index creation from existing documents. For example, if Porter stemming is used to build the index over a collection, then the same stemming method must be used to preprocess the query into terms.

A processed query contains a set of terms t_i that is used to query the built index. A query term might be restricted to particular features (i. e. $< feature, term >$ pairs). How the restriction of a search term to a particular feature of the document enhances the search process will be discussed in Chapter 7.

Example 2.5.4 Let us assume the sample user query uq_1:

$uq_1 \quad =$ "WNT in mouse"

If the query is more specific and the system allows to assign the feature "organism" to the query term "mouse" ($organism : mouse$), then the query can be refined into:

$uq1_{rev} = \{WNT, in, organism : mouse\}$

Preprocessing of $uq1_{rev}$ leads to the internal query presentation q_1:

$q_1 \quad = \{WNT, organism : mous\}$

Document retrieval The retrieval process results in a number of relevant documents $(d_i, \ldots, d_j)$ with respect to a given query q_i. To retrieve relevant models from the document collection, the query terms are mapped on the index terms. All relevant documents associated with those index terms are added to the result set.

Example 2.5.5 In the example, both documents d_1 and d_2 will be returned for the query q_1. d_1 contains the query term WNT. d_2 contains both query terms WNT and $mous$.

Result ranking A ranking of the result documents helps the user finding the best match for his query. A ranking function R (Definition 2.5.1) sorts a list of retrieved documents by their relevance with respect to a query q. The ranking function can be trivial to complex. The latter incorporate, for example, predefined weight concepts, user-defined weights, or relevance feedback methods in the ranking process. The choice of the similarity measure determines the result of the ranking process [Schmitt 2005] as well as the chosen IR model.

The VSM, for example, supports result ranking. Two standard methods to calculate similarity values between a query and a document (as well as between documents) are the *Scalar Product* and the *cosine similarity*.

A further concept in information retrieval are *weights*. They can give a feature a higher or lower importance, both in the query and in the result ranking. Weights can be applied during indexing or during query time. *Feature weights*, for example, are set manually during indexing. They weight a particular part of the document (e. g. "species name" encoded in an SBML file). *TF/IDF* assign further weights to terms based on the document and collection analysis. Term frequency (TF) refers to the number of times a given term t appears in a particular document d (term count), normalized by the number of terms in that document. Inverse document-frequency (IDF) refers to the general importance of a specific term within a particular collection; tf-idf is a common approach to determine term weights. Another type of common weights are *user given weights* allowing users to assign weights to particular query terms, for example through the search interface.

2.6. Summary

This chapter defined the basic terminology regarding modeling (model, simulation, experiment, bio-model), storage approaches for XML files, meta-information (data, information, knowledge), and information retrieval.

In this work, IR techniques extend the state-of-the-art SQL-based approaches to model search. The incorporated model features are multisided and necessitate a distinction between data, meta-data, information and meta-information. These terms have been defined.

The next chapter elaborates on the state-of-the-art of representation formats, meta-information encoding, and model storage.

3. State-of-the-art

People can't share knowledge if they don't speak a common language.

(Tom Davenport, Lawrence Prusak: Working Knowledge)

One important step towards enhanced model retrieval is a sophisticated bio-model search system. This can be facilitated by evaluating the meta-information encoded in state-of-the-art XML model representations. Model meta-information helps inferring knowledge on the Biology and assumptions behind the model and therefore enables more detailed queries on the underlying data.

Section 3.1 summarizes available meta-information for bio-model characterization. One concept for meta-information encoding are ontologies; a selection of existing biology ontologies is introduced in Section 3.1.2. The purpose of the *MIRIAM Guidelines* is to identify and informally describe relevant information about a model; the guidelines are described in Section 3.1.3. Section 3.2 then introduces approaches for the technical encoding of meta-information in SBML and CellML. Finally, Section 3.3 outlines storage solutions for XML-based model representation files.

3.1. Meta-information for computational biology models

The CSB community is aware the value of meta-information [Le Novère 2006; Le Novère et al. 2007; Wimalaratne et al. 2009b; Beard et al. 2009]. The CellML community, for example, states that

> there is an urgent need to [..] provide biological and biophysical annotation of the models in order to facilitate model sharing, automated model reduction and connection to biological databases. [Beard et al. 2009]

The needs for understanding existing models, reusing them in other context and extending them resulted in efforts concerned with the identification and encoding of relevant model meta-information. While the following section discusses the development process of generic *guidelines* for model annotation, the technical encoding of meta-information is described in Section 3.2.

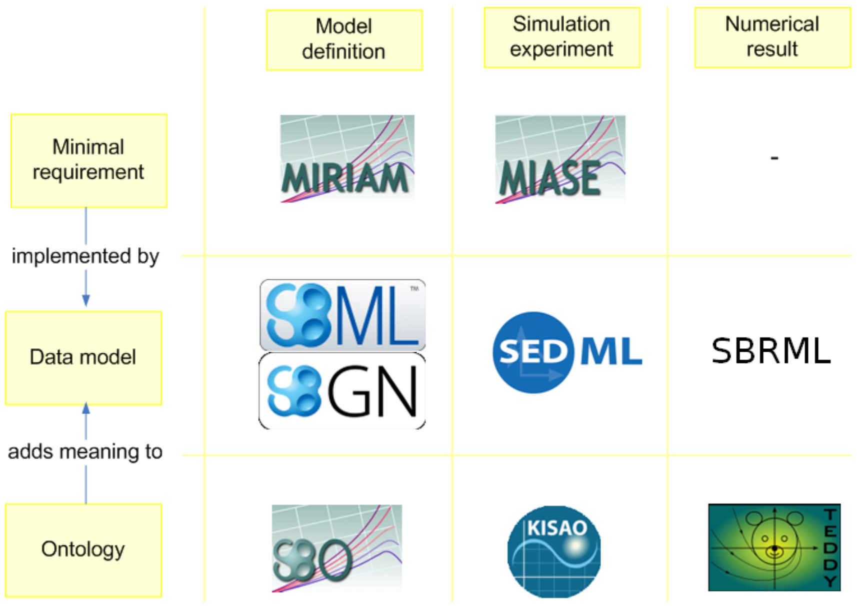


Figure 3.1.: Standard mosaic showing different kinds of meta-information usable for knowledge gain in the modeling and simulation of computational biological models. Adapted from [Chelliah et al. 2009], first presented in [Le Novère 2008]. The development of standards for simulation experiments is part of this thesis work.

3.1.1. The mosaic of standards

The community around SBML developed a set of standards for meta-information encoding. Chelliah et al. [2009] propose to organize them in the *mosaic of standards* shown in Figure 3.1.

On the data level (Figure 3.1, *data-model*), representation formats for model encoding (SBML, SBGN), for simulation experiment encoding (SED-ML, Section 5.4) and numerical result encoding (SBRML) are shown. The *Systems Biology Graphical Notation* (SBGN, [Le Novère et al. 2009]) is a standardized graphical notation for visualizing maps of biochemical and cellular processes in systems biology. The *Systems Biology Result Markup Language* (SBRML, [Dada et al. 2010]) is an XML-based format for the encoding of numerical results associated with a bio-model. SBRML aims at encoding and exchanging computational simulation results in a standardized way. It furthermore relates experimental data to a particular bio-model. One SBRML file may contain several data sets. Each data set contains a series of values associated with model variables, and their corresponding parameter values [Dada et al. 2010].

A common way of defining data formats for CSB is by first specifying their *Minimum Requirements*. MIs are federated in the aforementioned MIBBI project (Section 2.3.3). Most existing MI guidelines focus on the description of experimental studies. However, some also cover computational systems biology problems: The reference MI for SBML is MIRIAM (Section 3.1.3) and for SED-ML it is MIASE (Section 5.2).

The third level shown in Figure 3.1 reflects the specifically developed *ontologies* used in each data model to encode meta-information. For SBML, one such standard ontology is SBO (Section 3.1.2). SED-ML uses KiSAO (Sections 3.1.2 and 5.3), and SBRML considers the TEDDY ontology (Section 3.1.2).

3.1.2. Biology ontologies for model annotation

Due to the increasing complexity of developed bio-models, computer aid is needed to analyze the meaning, expressivity and behavior of a model. While the sole information about the model structure might be sufficient to build a model graph or to run it in a simulation environment, it is not sufficient to grasp a model's meaning. But as Le Novère et al. [2007] argue there is no point in exchanging data or models if nobody except the initial authors could understand the meaning of the data and the content of the models, respectively. In order to describe the biological phenomena under study, the model's XML representations are therefore additionally annotated, for example, to state that "this XML element 'stands for' the biological entity Drosophila". The models' mathematics is unambiguous (as it does not use words, but mathematical rules), but the *meaning* of the model and its biological and conceptual entities is not communicateable through the sole description of its mathematics. For the scope of this work, the meaning of an entity is regarded the "common features of the situations in which it [the entity] is used and of the activities which it produces" [Osgood et al. 1971, p. 2]. Ontologies are a suitable technology to encode this knowledge, and they have become an important concept for the annotation of bio-models (Section 2.3.2). Several communities are concerned with the development of ontologies to increase semantic content and standardization. One example is the *Open Biomedical Ontologies (OBO) foundry* [Smith et al. 2007]. It is a coordination body for the development and reformation of biomedical ontologies with regard to agreed upon principles for ontology design, particularly the ontology format, scope and standard relationships. As such it serves as an umbrella organization for several bio-ontologies. The following section describes some of the ongoing bio-ontology efforts.

Ontologies for biology entities

In 1998, the *Gene Ontology Consortium*[1] was founded to build the *Gene Ontology* (GO). It consisted of a group of database creators working in the field of bioinformatics. The initial goal of GO was to enhance the annotation of genes by a detailed structuring of related vocabulary. Today, GO is a community-developed, widely accepted controlled vocabulary [Ashburner et al. 2000]. It is made of three different so-called organizing principles. Those are *biological process* (referring to the question "What does the gene product do?", e. g. "cell division", `GO:0051301`), *molecular function* (referring to the question "How does the gene product act?", e. g. "insulin binding", `GO:0043559`), and *cellular component* (referring to the question "Where does the gene act?", e. g. "plasma membrane", `GO:0005886`)[2]. GO is constructed of directed acyclic graphs (i. e. hierarchical structures that allow for terms to have several parent terms). It has quickly developed into a widely-used tool for biologists and systems biologists and is now a *de facto* standard for gene annotation, used by many databases containing information about genes and proteins [Lambrix 2005]. The fact that it is considered a "community standard" and therefore heavily involves the users in its development process is seen as one of the reasons for the great success [Bada et al. 2004]. However, community efforts in the development of ontologies involve contributions from across various research groups, often resulting in terms that do not follow established conventions for the expression of concepts [Verspoor et al. 2009]. The outcome is a growing inconsistency in the concept naming and as well between similar terms, leading, for example, to redundant terms. Extra efforts have to be taken to assure the ontology quality (e. g. automatic term transformations) [Verspoor et al. 2009].

The *Chemical Entities of Biological Interest* ontology (ChEBI, [Degtyarenko et al. 2008]) focuses on the conceptualization of small molecules from the perspective of chemical entities. It is as well build of three different substructures, namely the *chemical entity* ontology (referring to physical entities of interest in chemistry, e. g. "ion", `CHEBI:24870`), *subatomic particle* ontology (referring to particles smaller than atoms, e. g. "proton", `CHEBI:24636`), and the *role* ontology (referring to a particular behavior shown by an entity, e. g. "enzyme inhibitor, `CHEBI:23924`). Besides the use of general OBO relationships (e. g. `is_a` or `part_of`), ChEBI developed a wide range of chemistry-specific relationships (e. g. `is tautomer of`).

GO and ChEBI each cover a small part of Biology. However, a huge number of further ontologies, controlled vocabularies, and databases for biological knowledge are

[1]`http://www.geneontology.org/GO`, last accessed 17 January 2011.

[2]See Janna Hastings, EBI Roadshow Workshop `http://www.ebi.ac.uk/training/roadshow/100924_rostock.html`, last accessed 14 March 2011.

available which concentrate on further aspects of Biology, such as protein sequences (UniProt, `http://www.uniprot.org/`, phenotypes (Human Phenotype Ontology, `http://www.human-phenotype-\discretionary{-}{}{}ontology.org`), anatomy (Foundational Model of Anatomy, `http://fma.biostr.washington.edu/`), or taxonomy (NCBI Taxonomy, `http://www.ncbi.nlm.nih.gov/taxonomy`). Many ontologies are devoted to a single organism or species, for example the *plant ontology* (`http://www.plantontology.org/`), or Flybase (`http://flybase.org/`).

Ontologies for biology modeling concepts

The *Systems Biology Ontology* (SBO, [Le Novère et al. 2007]) defines and relates terms for quantitative modeling and thereby fills the gap between the quantitative model (e. g. encoded in SBML) and the biological knowledge. For example, the semantic information provided by SBO enables developers to specify whether a species involved in a reaction was a simple chemical or a macro molecule, or what role it plays in the described process such as being an enzyme or an allosteric activator [Courtot et al. 2011]. SBO terms can be linked to bio-model constituents. In SBML, for example, the `sboTerm` attribute allows to explicitly put a model element into relation with an SBO term. SBO is build of six different vocabularies:

- the *modeling framework* branch (the assumptions underlying the mathematical description, e. g. "continuous framework", `SBO:0000062`),
- the *participant role* branch (function of a physical entity, e. g. "catalyst", `SBO:0000013`),
- the *entity* branch (a real being participating in an interaction or process, e. g. "enzyme", `SBO:0000014`),
- the *quantitative parameter* branch (a number representing an entities quantity, e. g. "Michaelis constant", `SBO:0000027`),
- the *mathematical expression* branch (the formal representation of a calculus linking parameters and variables, e. g. "Henri-Michaelis-Menten rate law", `SBO:0000029`), and
- the *interaction* branch (the action or influence happening to an entity at a given time or place, e. g. "ionisation", `SBO:0000209`).

SBO is part of the *biomodels.net* effort. The ontology is available for download from the bioportal[3] in OBO format. It is an integral part of SBML but it is also used by other communities such as CellML or SBGN.

[3]`http://purl.bioontology.org/ontology/SBO`, last accessed 12 December 2011.

The *Ontology of Physics for Biology* (OPB, [Cook et al. 2008]) covers terms that represent biophysical concepts and properties of anatomical entities. Doing so, it aims at bridging structural knowledge from the bioinformatics community with process knowledge from the biosimulation community [Cook et al. 2008]. OPB explicitly describes the mathematics and physics used in the bio-model encoding, referred to as "biophysical semantics of physics-based biosimulation models" [Cook et al. 2008]. It does, on the contrary, neither cover the representation of those entities (as provided by GO), nor their anatomies (as provided by the Foundational Model of Anatomy (FMA, [Rosse and Mejino 2003])). The theoretical basis is systems dynamics theory. OPB terms are to be used to annotate bio-model variables and equations. Cook et al. [2008] refer to cardiovascular models to exemplify the need for OPB: While many such models incorporate the blood pressure in the aorta, they use different variable names for it (e. g. "Paorta", "Pa", "X", or "Y"). To solve the problem of varying variable names in different models, a reference to the OPB term "Fluid pressure" (`OPB:OPB_00509`) can be used to characterize the physical entity represented by the variable. Another sample term from OPB is the *temperature dimension* as a *subclassOf* the *kinetic dimension.* OPB has been implemented in the *Protégé* ontology editor and is based on the OBO principles. It is available from the bioportal[4].

Ontologies for model-related information encoding

Behavior The *Terminology for the Description of Dynamics* (TEDDY, [Courtot et al. 2011]) provides terms for the systematic and machine-readable description of a model's observed behavior. It aims at covering the critical features of numerical results, both obtained from simulation and experimental measurements. The four branches of the TEDDY ontology cover vocabularies for concrete behavior, for diversification of behavior, for characteristics of behavior, and for functional motifs generating particular types of behavior [Courtot et al. 2011]. The *behavior characteristic* classifies the way that a dynamic system changes with respect to some environmental aspect (e. g. "exponential growth", `teddy:TEDDY_0000014`). The *Behavior Diversification* identifies behaviors based on quantitative properties (e. g. "Hopf Bifurcation", `teddy:TEDDY_0000072`). The *Functional Motifs* describe identifiable patterns in a (sub)model (e. g. "negative feedback", `teddy:TEDDY_0000034`). The *Temporal Behavior* characterizes a (temporal) sequence of states following the initial state (e. g. "Stable Limit Cycle", `teddy:TEDDY_0000114`). TEDDY is encoded in OWL and it is available from the bioportal[5]. It is also part of the biomodels.net effort.

[4]`http://purl.bioontology.org/ontology/OPB`, last accessed 12 December 2011.
[5]`http://purl.bioontology.org/ontology/TEDDY`, last accessed 12 December 2010.

Model simulation The *Kinetic Simulation Algorithm Ontology* (KiSAO, [Courtot et al. 2011]) is an ontology for the classification and characterization of kinetic simulation algorithms, mainly used in CSB. KiSAO has been developed during the course of this thesis; a detailed description is given in Section 5.3.

3.1.3. Minimum Information Required In the Annotation of Models

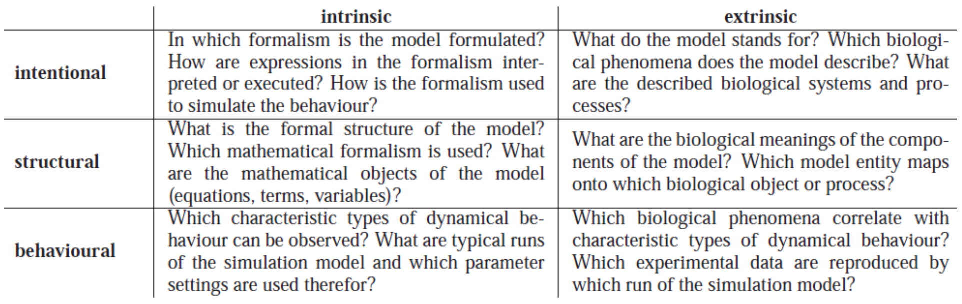
First ideas for standardized annotation of biochemical computational models were discussed during the October 2004 ICSB meeting. At the same time, Le Novère and Finney [2005] worked on a format proposal to encode meta-information for SBML models stored in BioModels Database. In 2005, the meeting manuscript and the proposal were both merged and published as the *Minimum Information Required in the Annotation of Models* (MIRIAM, [Le Novère et al. 2005]). Meanwhile, the MIRIAM guidelines have turned into a standard guideline for model annotation beyond the SBML community. They are also accepted by the CellML and NeuroML community, to mention a few. Lloyd et al. [2008], for example, state that they respect the MIRIAM framework in the model curation process.

MIRIAM defines the set of information to be provided about a model and its constituents in line with the model's publication. The goal of MIRIAM is to "define processes and schemes that will increase the confidence in model collections and enable the assembly of model collections of high quality" [Le Novère et al. 2007]. Information requested by MIRIAM includes the model author, the reference description, and information on the meaning of the model itself as well as of the model constituents. A model providing all MIRIAM-required meta-information is called *MIRIAM-compliant*. MIRIAM-required information can be divided into two parts [Le Novère et al. 2005]:

Standard for reference correspondence It ensures that the encoded model is associated with a reference description (i. e. a unique document describing it) and that the model is consistent with that description.

Scheme for encoding annotations It ensures the documentation of a model by external knowledge. This includes information required to associate the model with a reference description and the encoding process (*attribution annotation*) and the annotation with external data resources (*external data resources annotation*).

The MIRIAM *Standard for reference correspondence* comprehends the syntactic and semantic information about the model. It contains rules for the model encoding, the model structure and the results when instantiating the model in a simulation.



	intrinsic	extrinsic
intentional	In which formalism is the model formulated? How are expressions in the formalism interpreted or executed? How is the formalism used to simulate the behaviour?	What do the model stands for? Which biological phenomena does the model describe? What are the described biological systems and processes?
structural	What is the formal structure of the model? Which mathematical formalism is used? What are the mathematical objects of the model (equations, terms, variables)?	What are the biological meanings of the components of the model? Which model entity maps onto which biological object or process?
behavioural	Which characteristic types of dynamical behaviour can be observed? What are typical runs of the simulation model and which parameter settings are used therefor?	Which biological phenomena correlate with characteristic types of dynamical behaviour? Which experimental data are reproduced by which run of the simulation model?

Figure 3.2.: Meaning facets of a computational model of a biological system. Taken from [Knüpfer et al. 2006].

An example for rules about the *model encoding* is that the model must be encoded in a public, machine-readable format ([Le Novère et al. 2005, *rule 1*]). The requirement that a model structure must reflect the biological processes described in the reference description is part of the rules about the *model structure* (*rule 4*). Finally, the model should be made available together with all quantitative attributes that are necessary for the instantiation in a simulation (e. g. initial conditions and parameters) as defined in the rules for *results when instantiated in a simulation* (*rule 6*). The full list of MIRIAM compliance rules has been published in [Le Novère et al. 2005].

The *attribution annotation* associates a model unambiguously with a reference description to define its origin and the people involved in the model creation. Information on the encoding process has to be provided, including information on the model authors and creators. As part of the annotation scheme, the *external resource annotation* demands to provide meta-information about model constituents through links to corresponding structures in external data resources. To unambiguously link the external knowledge to a piece of model code, a so-called "triplet" consisting of {`data-type, identifier, qualifier`} is recommended (Section 3.2.1).

3.1.4. Further investigations: Meaning facets

Knüpfer et al. [2006] advocate *formal semantics for bio-models*. The authors claim that a formal representation for the simulation models and their mathematical explanations (intrinsic meaning) exists. The model's intended meaning (extrinsic meaning), however, is often described in natural language [Knüpfer et al. 2006]. Knüpfer et al. [2006] distinguish three levels of pragmatics which they identify as (1) the model's *intention*, (2) the model's *structure*, and the shown *behaviour*. In total, one arrives at six different meaning facets for a bio-model (Figure 3.2).

The discussion of meaning facets already led to the development of the TEDDY ontology (Section 3.1.2) which tackles the problem of encoding the behavior facet of a bio-model.

3.2. Meta-information encoding

The way meta-information is encoded in existing model representation formats is important for the subsequent information extraction. Ideally, there would be a common annotation scheme that allowed for abstracting from the underlying model representation format. However, the current situation is different: The representation formats introduced earlier (Section 2.2) either do not provide explicit containers for annotations, or they use their own specific ones. Methods to encode meta-information in SBML and CellML will be described in detail after a brief introduction to the MIRIAM reference standard.

3.2.1. The MIRIAM reference standard

MIRIAM-required meta-information should be encoded in triplets [Le Novère et al. 2005; Laibe and Le Novère 2007], referred to as the *MIRIAM reference standard* in the remainder of this work. The proposed annotation scheme follows the RDF idea (Section 2.3.4) and conceptually consists of a {`data-type,identifier,qualifier`} triplet [Le Novère et al. 2007] determining the predicate and object of a piece of meta-information:

data-type The data-type is a unique reference to a data resource.

identifier The identifier points to a particular piece of knowledge inside a particular data-type (i. e. both are connected).

qualifier The qualifier refines the relation between the annotated model constituent and the piece of knowledge associated to it.

MIRIAM recommends to provide the data-type as a *Unique Resource Identifier* (URI) (e. g. `urn:miriam:obo.go`) that can be resolved into a number of, for example, URLs (e. g. `http://www.geneontology.org/`). *MIRIAM Resources* [Laibe and Le Novère 2007] provide a repository of standardized URIs for a large set of different information sources[6].

The identifier is a particular entity from that data source (e. g. `GO:0016055` for the WNT receptor signaling pathway in Gene Ontology).

[6]`http://www.ebi.ac.uk/miriam/`, last accessed 28 July 2010.

The qualifier describes how the referenced piece of knowledge is related to the annotated model constituent (e. g. `occursIn`). A list of standardized qualifiers is available from the biomodels.net web site[7]. The two distinguished types of qualifiers are *model qualifier* relating to the model and the *bio-qualifier* relating to the biological meaning of a model constituent (including the model itself). An annotation of a bio-model representing the WNT receptor signaling pathway might be written in triplet form as {`urn:miriam:obo.go,GO:0016055,is-a`}.

3.2.2. Meta-information in SBML

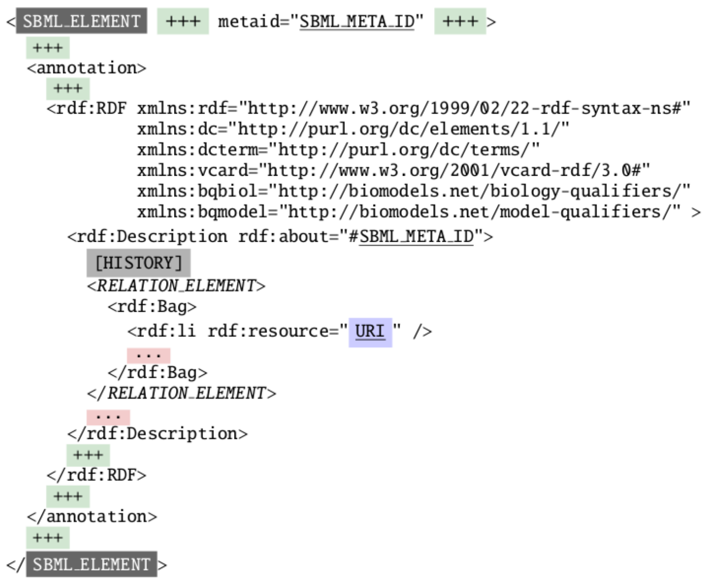
The current SBML Specification offers the `annotation` element for controlled model annotation. Furthermore, free-text meta-information can be provided with the model through the SBML `notes` element [Hucka et al. 2010]. SBO terms are stored in a specific `sboTerms` attribute.

The SBML annotation element In the current SBML version, meta-information is stored in the `annotation` element [Hucka et al. 2010]. The annotation of an SBML model and its constituents is optional. If it is present, however, it is suggested to follow the SBML annotation scheme described in [Hucka et al. 2010, Sec. 6.3]. SBML follows the proposed MIRIAM reference standard (Section 3.2.1); it uses RDF for the encoding of annotations and also reuses a subset of Dublin Core[8] and biomodels.net qualifiers.

An annotation must be embedded in an `rdf:RDF` tag (Figure 3.3). The annotated element is addressed through it's `metaid` inside the `rdf:about` attribute. The reference to third-party knowledge specifying the element's semantics must be placed inside the `rdf:Description`, more specifically must it be contained in an `rdf:li` list inside an `rdf:Bag` container. The link to an external resource must be perennial (i. e. consistently existing and not changing). To uniquely identify a controlled vocabulary term or object, the MIRIAM URI links to a physical source (i. e. a URL). The connection of the addressed third-party knowledge and the annotated element is established using any of the *bio-qualifiers* and *model-qualifiers* listed on `http://www.biomodels.net/qualifiers`. If an annotation follows the proposed scheme, it is considered an *SBML MIRIAM annotation.*

Listing 3.1 shows a sample annotation. Meta-information about the SBML species `Notch protein` is provided in this listing. It is annotated with an entry from the UniProt resource which is considered a homolog to the species itself (they have a common ancestor).

[7]`http://biomodels.net/qualifiers/`, last accessed 28 July 2010.
[8]http://dublincore.org/, last accessed 28 July 2010.



```
<SBML_ELEMENT +++ metaid="SBML_META_ID" +++ >
  +++
  <annotation>
    +++
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:dc="http://purl.org/dc/elements/1.1/"
             xmlns:dcterm="http://purl.org/dc/terms/"
             xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
             xmlns:bqbiol="http://biomodels.net/biology-qualifiers/"
             xmlns:bqmodel="http://biomodels.net/model-qualifiers/" >
      <rdf:Description rdf:about="#SBML_META_ID">
        [HISTORY]
        <RELATION_ELEMENT>
          <rdf:Bag>
            <rdf:li rdf:resource=" URI " />
            ...
          </rdf:Bag>
        </RELATION_ELEMENT>
        ...
      </rdf:Description>
      +++
    </rdf:RDF>
    +++
  </annotation>
  +++
</SBML_ELEMENT>
```

Figure 3.3.: The SBML standard annotation format. Taken from [Hucka et al. 2010]. Square brackets denote concepts (referring to further XML definitions), the string “. . . ” denotes zero or more elements of the same form as the preceding, the string “+++” denotes further valid XML content complying with the standard’s definition.

```
<species metaid="metaid_0000097" id="N" name="Notch protein"
         compartment="cytosol" initialConcentration="0.5"
         sboTerm="SBO:0000252">
 <annotation>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:dc="http://purl.org/dc/elements/1.1/"
           xmlns:dcterms="http://purl.org/dc/terms/"
           xmlns:bqbiol="http://biomodels.net/biology-qualifiers/"
           xmlns:bqmodel="http://biomodels.net/model-qualifiers/">
   <rdf:Description rdf:about="#metaid_0000097">
    [..]
    <bqbiol:isHomologTo>
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:uniprot:P46531"/>
     </rdf:Bag>
    </bqbiol:isHomologTo>
   </rdf:Description>
  </rdf:RDF>
 </annotation>
</species>
```

Listing 3.1: A sample SBML MIRIAM compliant annotation taken from the Goldbeter Pourquié model (BIOMD0000000201)

The SBML SBO attribute Specifically for the annotation of SBML elements with controlled vocabulary from SBO, SBML introduced the `sboTerm` attribute to the basic `SBase` class (Listing 3.1, l. 3). The `sboTerm` helps overcoming the problem that SBML `name` and `id` do not contain any semantic information. If an SBML element has an `sboTerm` assigned to it, the `sboTerm` defines the entity encoded by the SBML element. The SBML specification provides a mapping for the different SBML elements on main SBO identifier branches applicable for the annotation of that particular element [Hucka et al. 2010, p. 83].

In Listing 3.1, species `N` has an `sboTerm` assigned to it (`SBO:0000252`) which encodes the information that `N` is a "polypeptide chain". Further analysis of the SBO tree leads to the information that polypeptide chains are "macromolecules" (`SBO:0000245`).

3.2.3. Meta-information in CellML

Similar to SBML, CellML model representations do not capture biological information. On the contrary, the structure of a CellML model is far more general than the one of an SBML model (with CellML model elements "unit", "component", "connection", or "group" – opposed to the SBML "species", "compartment", "reaction" constructs). While this generalization allows for CellML models to be widely applicable, the language elements do express few biological information (e. g. they do not represent the entities and processes in the variable and component names) [Wimalaratne et al. 2009b]. Consequently, a thorough annotation scheme is crucial to understand and interpret CellML models. Especially with focus on multi-scale modeling issues, the CellML community emphasizes the need to further develop meta-data structures by reusing standards such as OWL and RDF [Beard et al. 2009]. CellML offers three different standard types for *annotations*: free-form comments of the person who coded the model into CellML (`comment`), a brief description of the limitations/scope of the content of the CellML element (`limitation`), and the description of the level of validation of the content of the CellML element (`validation`) [Cuellar et al. 2009].

While SBML considers all meta-data 'annotations', CellML distinguishes annotations from other meta-data and proposes two approaches to their encoding: the *CellML MetaData specification* [Cuellar et al. 2009; Beard et al. 2009] and the *CellMLBiophysical/OWL model* [Wimalaratne et al. 2009a,b].

The CellML Metadata specification is a labeling technique for CellML model elements using the RDF framework. Models are given semantic meaning by linking

CellML model elements to ontologies and controlled vocabulary such as SBO terms, BioPAX, UniProtKB or Gene Ontology entries [Lloyd et al. 2008].

The specification demands the description of a model's origin and creator, but also of the model itself and its constituents [Beard et al. 2009]. A special focus is set on model provenance, that is encoding information on the model development over time. CellML metadata aims at fulfilling the MIRIAM standard requirements, but still faces some problems. Particularly the assertion of fully resolvable models reproducing the results published in a reference description (as required by the MIRIAM reference correspondence [Le Novère et al. 2005]) and unambiguous annotation of each model constituent (as required by the MIRIAM external resource annotation [Le Novère et al. 2005]) need further work and practice by the CellML community [Beard et al. 2009]. The CellML community decided to use MIRIAM URIs to solve the latter problem; an example is given in Appendix A.3.3.

The namespace `"http://www.cellml.org/metadata/1.0#"` with prefix `cmeta:` was created for CellML metadata. Inside that namespace, different predefined *types* of meta-data exist, including `cmeta:modificationDate`, `cmeta:species`, or `cmeta:bio_entity` [Cuellar et al. 2009]. A recommendation for species meta-data is given in Listing 3.2. It shows the annotation of a CellML element with ID `#cellml_element_id` which relates the element to a species (`cmeta:species`) "Mammalia" and a species "Xenopus laevis".

```
<rdf:RDF
    xmlns:cmeta="http://www.cellml.org/metadata/1.0#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <rdf:Description rdf:about="#cellml_element_id">
  <cmeta:species>Mammalia</cmeta:species>
  <cmeta:species>Xenopus laevis</cmeta:species>
 </rdf:Description>
</rdf:RDF>
```

Listing 3.2: Recommended definition of species metadata. To be interpreted as: The element with ID `cellml_element_id` is relevant for all mammals and Xenopus laevis. Taken from [Cuellar et al. 2009, Fig. 18].

In addition to the meta-data specification, Nickerson and Buist [2009] propose the *CellML Simulation Metadata* (CMS) for the encoding of simulation setups. A third part of the metadata definition is the *CellML Graphing Metadata* (CGM) standard[9] which proposes a method to link simulation results from one or more performed experiments in order to provide two-dimensional graphs showing the model's behavior. Section 3.4 elaborates on the two standard proposals.

[9] `www.cellml.org/specifications/metadata/graphs`, last accessed 13 December 2010.

CellMLBiophysical/OWL model The CellML Metadata approach is not capable of expressing relationships between the biological processes and the annotated entities [Wimalaratne et al. 2009b]. As shown in Listing 3.2, a relation between the annotation and the annotated element is missing. The *CellMLBiophysical/OWL model* extends the aforementioned meta-data approach by structuring "the biophysical concepts captured in CellML models and allow modelers to explicitly annotate a CellML model with physical and biological information" [Wimalaratne et al. 2009b].

A CellMLBiophysical/OWL model is created by transforming the CellML XML representation into an OWL representation. An ontology has been developed for the representation of physical and biological processes described in existing CelLML models. The CellML/OWL model is annotated with biophysical concepts, in particular the CellML *components* and *variables* [Beard et al. 2009] using the so-called `cmeta:id` attribute. The created CellMLBiophysical/OWL representation is finally optimized using ontological mappings and graph reduction rules. The result is a simplified version of the CellMLBiophysical/OWL model representation of the original CellML model.

3.2.4. Annotations in use

Annotations are considered important and relevant by tool developers dealing with bio-models. A whole community working on "model semantics" (i. e. the encoding and processing of meta-information for bio-models) has evolved. Many efforts are closely related to SBML models. A comprehensive overview of SBML model annotation tools has been provided online by Frank Bergmann[10] (February 2010).

Tools for bio-model meta-information encoding A number of software tools are concerned with the automatic or semi-automatic annotation of bio-models. *SAINT* [Lister et al. 2009], for example, is a web application providing a "lightweight annotation integration environment". SAINT aims at overcoming the manual, time-consuming process of annotating a model, identified by the developers as one of the facts that hamper model reuse. The tool facilitates rapid and systematic model annotations and thereby helps in providing annotated quantitative bio-models to the CSB community. SAINT encodes annotations in the MIRIAM reference standard (Section 3.2.1). The annotation process is based on an integrated view of different data sources for model annotation, including *UniProtKB*[11], or *PathwayCommons*[12],

[10]`http://www.slideshare.net/fbergmann/`, last accessed 04 February 2011.
[11]`http://www.uniprot.org/help/uniprotkb`, last accessed 04 February 2011.
[12]`http://www.pathwaycommons.org/pc/`, last accessed 04 February 2011.

SBO, *Gene Ontology* and others. A query to SAINT is translated into single subqueries to the web services behind the above-mentioned data resources. As a result, SAINT suggests matching data, mostly ontology entries, for the automatic annotation of the model. The main focus is on the annotation of the model's entities (e. g. `species`). While SAINT has originally been developed to work with SBML models, ongoing investigations test its applicability on CellML models.

libSBMLAnnotation [Swainston and Mendes 2009] is a library with annotation functionality. On top of the manipulation functions for the reading and writing of annotations from and to (SBML) models provided by *libSBML* [Bornstein et al. 2008], the tool offers functionality to relate the annotations to external resource entries or among each other. One application of libSBMLAnnotation is the comparison of differently annotated model constituents in order to make a statement about their relation (e. g. finding identical or similar entities in two models that carry different entity names and annotations). libSBMLAnnotation uses the MIRIAM web service and incorporates cross-references provided by the single data resources. libSBMLAnnotation has been useful in the creation of large models, for example during the development of the genome-scale model of yeast metabolism[13] [Swainston and Mendes 2009].

The aim of *semanticSBML* [Krause et al. 2009] is to "help users to check and edit MIRIAM annotations and SBO terms in SBML models". The tool was designed to enable the merging of SBML models. Merges can be suggested and matching sub-models be identified based on the identified relations between the model constituent's annotations. SemanticSBML allows to edit and update model annotations graphically. It also supports the aforementioned MIRIAM reference standard. Based on query terms, possibly matching data source entries are suggested for annotation. External resources such as *KEGG*, *ChEBI*, or the *Gene Ontology* are incorporated in that matching process.

Tools for model transformation and visualization Annotations are furthermore used to visualize models and to perform transformations on them. One sample application is SBGN, a standard representation for biological network structures. SBO terms are used to annotate the SBGN graph[14]; those SBO terms are proposed to be used for the conversion of SBGN into SBML later on.

A second example for the use of annotations in model transformation tasks are the converters offered by BioModels Database to build different representation formats

[13]`urn:miriam:biomodels.db:MODEL0072364382`, last accessed 23 September 2010.
[14]A detailed example is given in [Juty et al. 2010].

from the existing SBML models[15] (e. g. CellML models).

Tools for model comparison and combination The aforementioned *semanticSBML* tool (formerly known as *sbmlMerge*) provides means for the semi-automatic merging of models. The mapping suggestions partially rely on the annotations. Also the already mentioned *libSBMLAnnotation* library finds corresponding entities in different models based on their annotations.

A second approach to using annotations for model comparison is the *Semantics of Biological Processes* (SemSim[16], last accessed 28 February 2011.) project. One of the sub-projects is concerned with annotating existing models of different formats and building an OWL representation of the semantic knowledge gained about those models. Based on that OWL model, tasks such as as model merging or sub-model extraction shall be enabled.

3.3. Storage of computational biology XML-models

The following first introduces approaches to bio-model storage (Section 3.3.1). Afterwards, a project concerned with model management issues in systems biology is described (Section 3.3.1). As model versioning is one specific aspect discussed in this work, solutions for bio-model versioning are then introduced in Section 3.3.2.

3.3.1. Model repositories

Published models are available from so-called *model repositories*. Among the best known ones are *BioModels Database* [Li et al. 2010], the *CellML Model Repository* [Lloyd et al. 2008], the *JWS Online Model repository* [Snoep and Olivier 2003], and *ModelDB* [Hines et al. 2004]. Those repositories will be introduced briefly, with special consideration of the chosen *storage approach* and provided *search possibilities*.

BioModels Database *BioModels Database* [Le Novère et al. 2006; Li et al. 2010] is a repository of freely accessible bio-models. It is an open-source project, and it is open to commercial and academic use. BioModels Database accepts models submitted by modelers (e. g. for reference in a publication) but also imports models from collaborative model repositories such as the CellML Model Repository. The main focus is on SBML-encoded models. For import of other formats a number of converters have been developed. The repository provides 269 curated and 361

[15]provided on the BioModels Database homepage `http://www.ebi.ac.uk/biomodels-main/`, last accessed 14 March 2011.

[16]`http://sites.google.com/site/semanticsofbiologicalprocesses/`

non-curated models with together hundreds of thousands of reactions[17]. BioModels Database provides several services, including model curation and annotations but also model presentation through a web interface, or model simulation through embedded simulation tools [Li et al. 2010]. The focus for this work, however, is on the model storage and search facilities.

Bio-models in SBML format are stored in two different ways: the quantitative information, kinetic laws and model entities are kept in the *SBML representation format* and the XML files are stored as such. Apache Lucene (`http://lucene.apache.org`) is used to index a subset of those model elements, and the index is kept for later search. The SBML file history is tracked using *Subversion*. Furthermore, the so-called *model metadata* is stored separately in a MySQL database. Metadata includes information on the submission and modification dates of a model file, authors' information, references and annotations encoded in the MIRIAM reference standard (MIRIAM URNs).

BioModels Database supports browsing and searching for models. Stored models can either be browsed from a list of available models (sorted by BioModels Database ID (BMID), model name, publication ID, or date of last modification)[18] or by using a tree-structured browser based on GO terms encoded in the models' annotations[19]. Search results are returned in an unordered result set. According to [Li et al. 2010], the so-called multi-step search system works in three sequential steps.Given a search term[20],

- First, the meta-data, publications and the annotations stored in the MySQL database are queried. The result of this search is a set of BMIDs.
- Secondly, the stored SBML XML files are queried, using the previously generated indexes, and parsing information such as the SBML `<notes>` tag. The returned BMIDs are added to the result set.
- If the search included query terms from external resources, then, thirdly, supplementary information is searched, either using information available in the local MySQL database, or web services. For the specific case of searching for a term in a taxonomy, the taxonomy tree is also traversed for neighbor terms, and model IDs associated with that term are additionally added to the result set.

[17]Last accessed 14 March 2011.

[18]`http://www.ebi.ac.uk/biomodels-main/publmodels`, last accessed 16 January 2011.

[19]`http://www.ebi.ac.uk/biomodels-main/modelstree`, last accessed 16 January 2011.

[20]The following is a simplified description of the search capabilities that does not distinguish between simple and advanced search.

The output is generated by using the BMIDs to query the MySQL database for the formerly extracted meta-data that is necessary for display on the web site.

The CellML Model Repository The *CellML Model Repository* [Lloyd et al. 2008] is an online repository for CellML models at different stages of curation. It is an instance of the *Physiome Model Repository 2* (PMR2), a model management system based on the Content Management System *Plone*[21].

The available models cover a wide range of different biological processes, among others signal transduction pathways, metabolic pathways, electrophysiology, immunology, cell cycle, muscle contraction and mechanical models [Lloyd et al. 2008]. It is the intention of the maintainers of the repository to bring forward the model curation and annotation process so that ideally all models "replicate the results in the published paper" and the search for models and elements within models is facilitated. The *CellML model repository* contains about 500 model exposures[22] encoded in the CellML format. An exposure is a model and its associated documentation and meta-information.

Models in the CellML Model Repository are browsed by different (physiological) categories, including cell cycle, signal transduction, or metabolism. A CMS-wide full-text search is offered that allows for simple free text search. Additionally, models of different curation states can be searched. A search by particular model features (e. g. specifically by author or publication year) is not possible. Search results are returned in an unordered result set.

ModelDB *ModelDB*[23] [Hines et al. 2004] is a database for curated models related to computational neuroscience and independent of a particular model encoding format. The design of ModelDB focused on providing authors a repository for the storage of models, in particular in preparation for submission in neuroscience journals. ModelDB is capable of storing models "encoded in any language for any environment" [Hines et al. 2004]. It stores the originally submitted model files, that is the complete code specifying the attributes of the original biological system represented in the model, including interface and control code to run the model in the associated simulation environment, and a non-standardized readme text file explaining briefly how to use the provided computer code [Hines et al. 2004]. The submitted code should contain the biological entities (e. g. `neuron` or `synapse`) but also used concepts such as `synaptic plasticity` or `pattern recognition`, and

[21] `http://plone.org`, last accessed 14 March 2011.
[22] Last accessed 02 March 2011.
[23] `http://senselab.med.yale.edu/modeldb/`, last accessed 14 March 2011.

the software used to implement the model (e. g. `NEURON` or `XPP`). Secondly, ModelDB stores model meta-information which is incorporated in the model search. This information includes a concise statement of the model purpose, how to use it, and a complete citation of the reference publication [Hines et al. 2004].

The underlying database management system is Oracle 10[24]. ModelDB is an instance of the *Entity–Attribute–Value/Classes–Relationship* framework (EAV/CR, [Marenco et al. 2003]) for data representation (also referred to as "open schema"). The *EAV* approach originated from the context of artificial intelligence where it was developed as an information representation format: Data are stored in a single table with three columns, namely the *entity* (i. e. the object being described), an *attribute* (i. e. an aspect of the described entity), and the atomic *value* of that attribute [Marenco et al. 2003]. The approach is similar to the generic storage approach for XML documents. In addition, the *CR* approach then combines object-oriented concepts with the EAV concept. Each object of the database must belong to a particular *class*, and values may be other objects. In that way, *relationships* between objects are defined. The EAV/CR concept is also used to build the RDF statement structure introduced earlier (Section 2.3.4). An example for such EAV/CR triple as stored in ModelDB is *Neuron* (entity) – *Soma_location* (attribute) – *ParsCompacta, S.Nigra*[25] (value).

The search functionality in ModelDB relies on the meta-information entered by the model submitter. Search by author name or accession number (modelDB ID) are supported. The complete list of models can be returned sorted by the model name or by the author. Additionally, some predefined queries regarding different criteria such as *cell type* or *simulators* are available. However, the queries do not incorporate the model files themselves; as such a search on the model code is not possible. The meta-information is not standardized, but consists of partially predefined strings and partially manually entered data. Third-party knowledge is not incorporated in the search process; the submitted models are not annotated.

JWS Online – Model Database The *JWS Online – Model Database*[26] is part of the *JWS Online Simulator* [Olivier and Snoep 2004], a web-based simulator for biochemical kinetic models. The model repository serves as the maintainer for a number of kinetic models that can be interactively run online. It supports the search 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

[24]from e-mail correspondence, 22 September 2010.
[25]The *S.Nigra* is a brain structure located in the mid brain.
[26]`http://jjj.biochem.sun.ac.za/database/`, last accessed 14 March 2011.

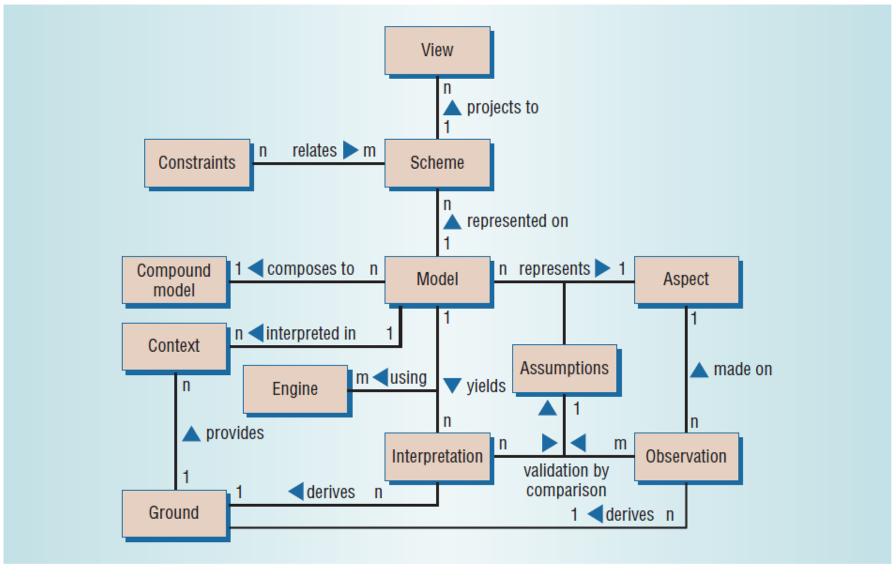


Figure 3.4.: Systems biology meta-model identifying key concepts in systems biology and their relationships. Construction: model, compound model, scheme, constraints, view components. Analysis: model, context, engine, interpretation, ground components. Validation: model, aspect, observation, assumptions, interpretation components. Taken from [Finkelstein et al. 2004].

searchable categorization of models in the repository, distinguishing, for example, between "cell cycle" models and "metabolism". A full text search is not supported. Search results are returned ordered by author name. As there does not exist a publication on the technical background of the model repository, further information on the backend of the provided interface cannot be given.

The UCL Beacon Meta-model A general study of the model management issues for Systems Biology models [Finkelstein et al. 2004] has been provided in line with the *UCL Beacon project* (`http://grid.ucl.ac.uk/biobeacon/`). The motivation behind this work was to enable the indexing, comparison and integration of diverse computational systems biology models. The two main suggested concepts for use in systems biology were *simplification* which allows to abstract from the biological complexity and *modularity* which is required to enable the break up of complex systems into manageable components which later on can be reassembled [Finkelstein et al. 2004].

A model-centric storage approach for computational models has been developed to realize the project goals. It encodes three information regions necessary in systems biology modeling: (model) construction, analysis and validation. The meta-model consists of 12 connected entities and their attributes (Figure 3.4). Only the following subset of entities is relevant for this work [Finkelstein et al. 2004]:

model A model is a standardized file with a certain underlying scheme and a version (e. g. “Biomodels Model 21” in SBML Level 2 Version 1 format).

scheme A (representation) scheme corresponds to the language that the model is constructed in (e. g. “SBML Level 2 Version 1”).

aspect Models represent particular biological aspects such as ”system description” (i. e. properties of biological interest that can be seen as labels for a model). Aspects are described through ontology references which are recorded in a special *origin* entity and include experiments, publication references, etc.

interpretation Models can be instantiated and then yield interpretations through analysis (e. g. simulation or static analysis). The results and conclusions gained from the model are stored in the interpretation entity.

observations Biology experiments lead to observations about a phenomenon; they enable the development of hypotheses about a biological model, but also validation of interpretations derived from those models.

context The context entity refers to data that is required to interpret the model (i. e. the model’s input). It holds all information about the model parametrization, including the values and their assigned confidence.

engine For each model there is a number of engines defined which describe how to execute the model to yield a particular interpretation. Engines include simulation tools, computer programs, or humans. One example for such an engine is the *COmplex PAthway SImulator* (COPASI, [Hoops et al. 2006]).

The disregarded entities (e. g. `assumptions`, `view`, `constraint`, or `ground`) only play a minor role in the following work; they are described in [Finkelstein et al. 2004].

A subset of the identified entities are considered in the model database design introduced in Section 6.3.3. A second subset is relevant for the development of the simulation experiment description format introduced in Section 5.4. In contrast to this work, the UCL beacon project does not investigate search and retrieval methods for the stored data. The UCL beacon project focuses on model management issues. It concentrates on the model *surroundings* rather than on the fine-grained definition of model constituents. Furthermore, it deals with model-related issues, including related experiments, findings and assumptions during the modeling.

3.3.2. Model versioning

Models may be produced in different versions over time, and by different research groups; furthermore, researchers may produce a number of instances of a model, using different configurations [Finkelstein et al. 2004]. One also has to distinguish versioning *inside* the model files from versioning provided by the *model repository*. Versioning support has been discussed in both, the representation format communities, and among the model repository developers. However, model versioning tools for the user are at preliminary stage.

Another problem, not addressed in this thesis though, is the versioning of bio-ontologies. When working with meta-information based on ontology references, it is important to keep track of ontology evolutions and ensure that the annotations are valid and consistent. However, the good practice with bio-ontologies is to keep all term identifiers, but only mark them “obsolete” if necessary.

Versioning support in representation formats

The CellML community discusses the importance of thorough versioning concepts for the model reuse process [Beard et al. 2009], saying that the current CellML metadata specification allowed for detailed revision histories to be associated with a model. Beard et al. [2009] distinguish “trivial changes” (e. g. error corrections during the translation of a model to CellML) from “substantial” ones (e. g. creating revisions of a model), but for any changes applied to a model they demand “these changes are listed and fully documented so that a prospective user knows what has been changed and why, by whom and when”. As an example for the usefulness of model versioning the publication mentions the history of parameter value changes.

SBML provides a model history concept through additional elements for the description of modifications in an SBML model and its constituents [Hucka et al. 2010, Sec. 6.6]. The history is restricted to the SBML encoding and does not cover conceptual changes in the model. The main points 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). The model history is embedded in an SBML file as part of the SBML meta-information encoding (Figure 3.3 on page 53). It might occur in any model constituent, being defined in an `rdf:Description` block inside the `annotation` element. The history always refers to a piece of SBML code addressable by an `SBML_META_ID`. Figure 3.5 shows the definition of the *model history* in detail [Hucka et al. 2010, p. 89]. The history encodes information on the creator of an annotation using the Dublin Core

```
<dc:creator>
  <rdf:Bag>
    <rdf:li rdf:parseType="Resource">
      +++
      <vCard:N rdf:parseType="Resource">
        <vCard:Family> FAMILY_NAME </vCard:Family>
        <vCard:Given> GIVEN_NAME </vCard:Given>
      </vCard:N>
      +++
      [<vCard:EMAIL> EMAIL_ADDRESS </vCard:EMAIL>]
      +++
      [<vCard:ORG rdf:parseType="Resource" >
        <vCard:Orgname> ORGANIZATION_NAME </vCard:Orgname>
      </vCard:ORG>]
      +++
    </rdf:li>
    ...
  </rdf:Bag>
</dc:creator>
<dcterms:created rdf:parseType="Resource">
  <dcterms:W3CDTF> DATE </dcterms:W3CDTF>
</dcterms:created>
<dcterms:modified rdf:parseType="Resource">
  <dcterms:W3CDTF> DATE </dcterms:W3CDTF>
</dcterms:modified>
...
```

Figure 3.5.: The SBML history and associated elements. Taken from [Hucka et al. 2010, p. 89]. Square brackets denote concepts (referring to further XML definitions), the string "..." denotes zero or more elements of the same form as the preceding, the string "+++" denotes further valid XML content complying with the standard's definition.

`dc:creator`, including the *name*, *email-address* and *organization name*[27]. That information must be followed by the `dc:terms:created` element that provides the creation date of that model constituent. Then a list of `dcterms:modified` elements contains the modification dates. Dublin Core makes use of the W3C date format for the encoding of that time information.

Versioning support in model repositories

The BioModels Database team intends to investigate a sophisticated versioning system for SBML models [Li et al. 2010]. The aim is to allow users to retrieve and compare different versions of a model and its annotations. BioModels Database to date provides a bio-model in different encodings, including the various levels of SBML and automatically generated format such as CellML. Internally, the SBML model version at each release is kept in an SVN repository. However, the particular SBML file in its different versions is not available for the users. An exception is the link to the original file, which is the first uploaded version of the model in its original format. Changes that might have been applied to the model during its existence in

[27]In SBML, the creator is referred to as the person who created the SBML encoding of the referenced model constituent.

the model repository are not traceable.

ModelDB provides preliminary version support for a subset of the available models[28] through an instance of the distributed version control system *Mercurial*[29]. The system provides a list of versioned model, sorted by accession number. A more detailed log is available for each model, including the age of each particular version available for that model, the version submitter, and a short description of that version, similar to an SVN comment. The log can be visualized in similar ways, including a tag-based view, a graph-based view, or a branch-based view. Finally, the change set between two model code versions can be determined, using a standard diff algorithm.

Similar to ModelDB, CellML uses a *Mercurial repository* to keep track on different model versions. Models can be stored locally in a version-controlled Mercurial instance, a so-called *model workspace* [Magjarevic et al. 2009]. Once submitted to the CellML Model repository, those models can then be imported as a particular revision of a workspace. The workspace revisions also enable authors to review the change sets of their models. The way that Mercurial is used in PMR2 allows not only for the maintenance of models but of any generic file type (e. g. experimental setups, docs, or simulation experiment descriptions).

3.4. Simulation description formats

The necessity for storing simulation experiment descriptions led to the development of formats for the description of simulation experiments – both for the field of CSB, and for simulation experiments in general.

Some of the efforts existing in CSB are limited to particular simulation tools and serve as internal storage formats rather than broad exchange formats. One example is the internal XML format used by the simulation tool COPASI [Hoops et al. 2006]. It allows to export and import simulation experiment descriptions run on a model inside COPASI.

The CellML community developed its own format that is independent of the simulation tool, called the *Simulation Metadata Specification* [Miller 2009]. In using RDF, it follows the style of CellML model development and meta-data annotation. The meta-data may not only be included in the model specification but can also be stored in a separate document. An example for simulation meta-data specified inside a CellML model is given in Listing 3.3.

```
<model name="CoupledPendulum_version01" [..]
```

[28]`http://neuro.med.yale.edu/hg`, last accessed 14 March 2011.
[29]`http://mercurial.selenic.com/`, last accessed 14 March 2011.

```
2        cmeta:id="CoupledPendulum_version01">
3  <rdf:RDF [..]>
4   <rdf:Description rdf:about="">
5    <cs:simulation rdf:parseType="Resource">
6     <cs:simulationName>SwingFor100s</cs:simulationName>
7     <cs:multistepMethod>implicit-runge-kutta-2</cs:multistepMethod>
8     <cs:linearSolver>direct</cs:linearSolver>
9     <cs:variablesImportantInSimulation rdf:parseType="Collection">
10     <rdf:Description rdf:about="#time" />
11     <rdf:Description rdf:about="#a_angle" />
12     <rdf:Description rdf:about="#b_angle" />
13    </cs:variablesImportantInSimulation>
14    <cs:boundIntervals rdf:parseType="Collection">
15     <rdf:Description>
16      <cs:boundVariable>
17       <rdf:Description rdf:about="#time" />
18      </cs:boundVariable>
19      <cs:maximumStepSize>1</cs:maximumStepSize>
20      <cs:tabulationStepSize>0.1</cs:tabulationStepSize>
21      <cs:startingValue>0</cs:startingValue>
22      <cs:endingValue>100</cs:endingValue>
23     </rdf:Description>
24    </cs:boundIntervals>
25   </cs:simulation>
26  </rdf:Description>
27 </rdf:RDF>
28 </model>
```

Listing 3.3: Extract from a CellML model including simulation meta-data following the suggestions by Miller [2009]

The format proposal consists of the following parts (line numbers refer to the example given in Listing 3.3):

- a simulation reference from inside the model description (`cs:simulation`) with an optional name (ll. 6-7),
- a description of the linear solver (`cs:linearSolver`) (l. 9), an iteration method (`cs:iterationMethod`), or a multistep method (`cs:multistepMethod`) (l. 8), and
- the definition of bound variables and the boundaries (`cs:boundIntervals`) (ll. 15-25).

The specification is used for the description of simulation experiments on CellML models only. Nickerson et al. [2008] further state that the format is intended for the description of simulation meta-data for *electrophysiological models*. Apart from that restriction, a second disadvantage of the CellML meta-data approach is that the description is tied to one particular model which is referenced through the `rdf:Description about="`*`model`*`"` element (l. 6). It is in addition not possible to specify model perturbations within the simulation description. Last but not least, the information about the solver and iteration method are stored as string chains in an XML element, instead of using controlled vocabularies.

The JAMES II framework [Himmelspach et al. 2008] uses an internal, but very generic format to describe simulation experiment descriptions. The demands for the description of a JAMES II experiment include an unambiguous model identification, the separation of model and experiment, the consideration of different model formalisms (i. e. modeling frameworks) and programming languages, as well as means for the configuration of an experiment [Himmelspach et al. 2008]. The *ExML* format [Oertel 2009] is a proposal for a generic XML exchange format aiming at fulfilling the demands mentioned above. Due to its generality it is applicable to a wide range of simulation experiments. The structure of an ExML encoded experiment description separates *model description* and *experiment description*. The model description includes the model definition and system data (i. e. data related to the model and necessary for further applications), for example parameter value updates or references to validation data sets. The experiment description contains the complete set of experiments applied to the model, and their description [Oertel 2009]. To the authors knowledge, ExML is so far not supported by existing software.

An investigation for the development of a programming language like format for the description and conduction of simulation experiments is Simplex-EDL [Leh 2008] which is used as the internal format in the Simplex III simulation environment.

3.5. Summary

In this chapter, I explained ongoing works related to the meta-information encoding of computational biology models, thereby, for the first time, desribing meta-information related efforts to model reuse in a structured and comprehensive manner.

A number of different standards evolved over the last years, including Minimum Information guidelines, formats for the technical encoding of the different information, and ontologies to describe the encoded knowledge in a standardized way. I introduced MIRIAM which describes the meta-information that should be delivered with a model. The technical approaches to encoding that meta-information in two model representation formats, SBML and CelLML, have been discussed in detail. Bio-ontologies play an important role in representing the biological knowledge in a computer-readable and stable manner. The relevant ontologies for this work have been introduced and characterized. The state-of-the-art in bio-model storage has been illustrated with particular focus on the major repositories (BioModels Database, CellML Model repository, JWS database, and ModelDB). Finally, preliminary work on the encoding of simulation experiments has been introduced.

I conclude that the next step towards better model reuse is the development of a generic retrieval system applicable to the introduced model repositories as current systems lack sophisticated search and versioning capabilities. Furthermore, the standardized encoding of simulation experiments is necessary to incorporate information on applicable simulation runs in the model retrieval process and to enable faster simulation of retrieved models.

The following conceptual works are based on the just introduced approaches; parts of the state-of-the-art methods are extended and improved.

4. SBML meta-information encoding

'When I use a word,' Humpty Dumpty said, in a rather scornful tone,' it means just what I choose it to mean, neither more nor less.'

(Alice in Wonderland)

Model reuse postulates the definition of an expressive and correct format for meta-information encoding.

Here I propose a format for meta-information encoding in SBML models. Section 4.1 summarizes the known problems and shortcomings with current SBML annotations. Section 4.2 then introduces the enhanced meta-information encoding approach. Finally, Section 4.3 summarizes the results and discusses the applicability of the proposed annotation standard on other representation formats using the example of πML.

The described proposal has been advocated to the SBML community as the *Annot package extension to the SBML Level 3 Core* [Waltemath et al. 2011d]. The approach is also applicable to other bio-model representation formats.

4.1. Problem statement

Existing model representation formats use their own annotation schemes. Although the MIRIAM standard is widely accepted for model annotation requirements, the syntactical encoding of meta-information is not standardized. Standardized meta-information encoding, however, is a prerequisite for comparing models on the annotation level regardless of the underlying modeling approach.

SBML is the main standard for the encoding of bio-models (Section 2.2.1). Already in 2006, the annotation standard for SBML models was suggested (Section 3.2.2) and soon adopted by software tools. Figure 3.3 on page 53 shows the structure of the current SBML annotation standard. Over the last years, the SBML annotation scheme has been embraced by a variety of tools to encode, process and visualize semantic information about a bio-model [Krause et al. 2009; Schulz et al. 2010; Lister et al. 2009; Swainston and Mendes 2009; Henkel et al. 2010]. The evaluation of annotated models, however, showed some weaknesses regarding the encoding and interpretation of annotations. It furthermore lacks expressiveness of complex relations such as nested and listed annotations. The specification is not detailed enough and leaves too much room for interpretation, as well as misinterpretation. With the progress in modeling biological systems and the models' increasing complexity, more information than currently possible needs to be encoded. This section depicts the identified problems with the current SBML annotation scheme described in the SBML L3 specification [Hucka et al. 2010]. Section 4.2 will afterward propose solutions.

Container and collection support The SBML standard annotation supports the RDF container `rdf:Bag` only. It does not approve `rdf:List`, `rdf:Alt` and `rdf:Seq` (Section 2.3.4).

Negations of annotations Using RDF, constituents can be described through their specified characteristics. However, *exclusion* of particular characteristics is not supported. It is as well impossible to define closed lists (`rdf:List`) to implicitly make negative statements, as SBML only allows using the `rdf:Bag` container.

RDF/XML centric view The current SBML annotation standard takes a syntax-oriented approach to describing the annotation scheme; its capabilities are defined on RDF/XML level. However, RDF offers different ways of modeling statements (Section 2.3.4). Modeling on the RDF graph level is robust and unambiguous. One RDF graph can be represented by several different RDF/XML notations. Hence current SBML RDF statements cannot be modeled on RDF graph level as the SBML annotation-compliant serialization into a syntactical representation cannot be assured. Defining the expressiveness of annotations on RDF/XML level restricts the users to that particular notation and therefore disregards many RDF tools.

Qualifying relations The qualifying relations used in the SBML annotation scheme are limited to bio- and model-qualifiers (Appendix A.1). The RDF primer [Miller and Manola 2004] suggests the naming of subject, predicate and object to be considered as *parts*. For example, the subject is defined as "the part that identifies the thing the statement is about" [Miller and Manola 2004, Sec. 2.1], and the predicate is defined as "the part that identifies the property or characteristic of the subject that the statement specifies". A typical RDF triple, or statement, consisting of {`subject, predicate, object`} then reads: "`subject` has a `predicate` whose value is `object`". The current naming of bio-qualifiers (e. g. `is`, `isVersionOf`, isHomolog) and model-qualifiers (e. g is, isDescribedBy) does not easily allow to formulate the above statements. The main problem is that the names do not refer to nouns but are verbs.

Missing attribute annotations The current annotation scheme takes the RDF approach to providing rdf:Descriptions for SBML XML elements such as for species or compartment, but it lacks a mechanism to annotate SBML attributes such as the initial concentration of the species (Listing 4.1).

```
 1 <species metaid="metaid_0000042" id="Y" name="Intravesicular Calcium"
 2          compartment="intravesicular" initialConcentration="0.36">
 3  <annotation>
 4   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 5            xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
 6    <rdf:Description rdf:about="#metaid_0000042">
 7     <bqbiol:is>
 8      <rdf:Bag>
 9       <rdf:li rdf:resource="urn:miriam:obo.chebi:CHEBI%3A29108"/>
10       <rdf:li rdf:resource="urn:miriam:kegg.compound:C00076"/>
11      </rdf:Bag>
12     </bqbiol:is>
13    </rdf:Description>
14   </rdf:RDF>
15  </annotation>
16 </species>
```

Listing 4.1: SBML species element with annotation. Namespaces partially removed. Example extracted from urn:miriam:biomodels.db:BIOMD0000000100.

Both annotations in the rdf:Bag (ll. 8-11) describe the species instance (ll. 1-2) as a whole. However, it is impossible to specify that an annotation is particularly denoted to an SBML element's attribute (e. g. compartment).

Annotations about annotations To date annotations about annotations are not supported. For example, the person who provided a particular annotation cannot be encoded (e. g. rdf:Description rdf:about="#metaid_0000042", ll. 6-13 in Listing 4.1). It is also not possible to specify who last modified a particular annotation and why. The general problem is that the current SBML annotation scheme only supports a subset of RDF which excludes identifiers on the rdf:Description elements. Consequently, annotations can only be related to SBML elements (by referring to the SBML elements' metaid) but not to other annotations.

Relating several qualified annotations Elements having a bio-qualifier or model-qualifier as their predicate inside the rdf:Description are referred to as "qualified annotations". With all qualified annotations for a model constituent being at the same level (Listing 4.1, ll. 8-11), it is currently impossible to define relations between them. A nesting of qualified annotations is not supported (e. g. "species `#metaid_0000042 bqbiol:is [A and (B OR C)]`"). However, the nesting determines the biological meaning of annotated constituent.

4.2. Enhanced SBML annotations: Annot package

The following sections summarize the main solutions to the problems introduced in Section 4.1. Namespace declarations are suppressed. The complete documentation is available from [Waltemath et al. 2011d].

4.2.1. Package scope and integration into SBML

Due to the modular nature of SBML L3, package definitions can extend the core language definition. One such extension is the Annotation package definition which proposes to expand the current `<annotation>` element. The Annotation package defines the encoding of annotation information inside SBML models. SBML recommends a namespace scheme for packages[1]; the Annotation package namespace follows this standard and is `http://www.sbml.org/sbml/level3/version1/annot/version1`.

Two proposals were made originally for the use of the new annotation scheme in SBML: The first attempt was to use a new `<annot:annotation>` element inside the current `<annotation>` element, but outside the current `<rdf:RDF>` block as a subelement of `annotation`. However, a new `<annot:annotation>` element has finally been defined as a sibling of the existing `<annotation>` element (Listing 4.2). This solution provides a much clearer distinction between the package extension elements and the core SBML file. Listing 4.2 shows the use of the Annotation package in an SBML file.

```
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3"
    version="1"
     xmlns:annot="http://www.sbml.org/sbml/level3/version1/annot/version1
         "
     [..]
     annot:required="false" [..] />
[..]
 <model>
  <listOfReactions>
   <reaction id="Reaction1" metaid="_905823" name="degradation of LacI
       transcripts" reversible="false" sboTerm="SBO:0000179">
    <annotation>
     <rdf:RDF>
```

[1]http://sbml.org/Community/Wiki/SBML_Level_3_Core/Package_mechanism, last accessed 21 October 2010.

```
11       [SBML STANDARD ANNOTATION]
12      </rdf:RDF>
13     </annotation>
14     <annot:annotation>
15      <rdf:RDF>
16       [ANY RDF SPECIFIED IN THE ANNOT PACKAGE]
17      </rdf:RDF>
18     </annot:annotation>
19     <listOfReactants />
20     <kineticLaw />
21    </reaction>
22   </listOfReactions>
23  </model>
24 </sbml>
```

Listing 4.2: Integration of `<annot:annotation>` annotations in SBML

Specific to the Annotation package are (1) the namespace declaration (l. 2), and (2) the declaration of the `annot:annotation` element as optional (l. 4). An SBML model can correctly be parsed and executed without the information encoded in Annotation package. The `annot` element is therefore optional. This is indicated by adding the XML attribute `annot:required` to the `<sbml>` element and setting its value to `false` (l. 4). Each SBML element may then have SBML standard annotations assigned to it (ll. 9-13), but also annotations from the *annot* namespace (ll. 14-18). The package definition does not depend on other packages.

4.2.2. Qualifier names

To comply with the RDF scheme of subject-predicate-object triples where all parts are nouns (read: "SUBJECT has PREDICATE who's value is OBJECT") the existing qualifiers have been updated. While predicates in the past have been verbs or word groups starting with a verb, all predicates were transformed into nouns. For example, the former bio-qualifier "bqbiol:is" was converted into "bqbiol:identity" (Appendix A.1).

4.2.3. Container and collection support

To enable the definition of closed lists, ordered annotations and alternative annotations, the Annotation package extends the current list of supported containers (i. e. `rdf:Bag` elements) by `rdf:List`, `rdf:Seq` and `rdf:Alt` elements.

SBML annotations to date imply (implicitly) an *alternative* relationship between two or more annotations inside an `rdf:Bag` (Listing 4.1, ll. 8-11). The `rdf:Alt` container as proposed by the Annotation package now allows to explicitly say that an SBML element can have alternative annotations. Similarly, the `rdf:List` allows to explicitly provide a closed list of valid annotations, and the `rdf:Seq` allows to define a sequence of elements.

Listing 4.3 exemplifies the application of `rdf:Alt` containers: The species with meta-id `meta_glc` may be annotated as `urn:miriam:obo.chebi:CHEBI%3417234` or as `urn:miriam:kegg.compound:C00234`. Both annotations are alternatives and as such valid.

```
<species id="glc" metaid="meta_glc" name="Glucose">
 <annot:annotation>
  <rdf:RDF>
   <rdf:Description rdf:about="#meta_glc">
    <bqbiol:identity>
     <rdf:Alt>
      <rdf:li rdf:resource="urn:miriam:obo.chebi:CHEBI%3417234"/>
      <rdf:li rdf:resource="urn:miriam:kegg.compound:C00234"/>
     </rdf:Alt>
    </bqbiol:identity>
   </rdf:Description>
  </rdf:RDF>
 </annot:annotation>
</species>
```

Listing 4.3: Use of RDF containers in the Annotation package. Taken from [Waltemath et al. 2011d].

4.2.4. Attribute annotations

To support attribute annotations, the Annotation package proposes to extend the current reference system to distinguish SBML element references from SBML attribute references. Examples for the specific attribute annotation are providing the information (1) who set up an initial concentration value (opposed to providing information on who defined the species) or (2) why an attribute has a given value. The first example demands a reference to a person and the second one needs a link to an external publication describing the value.

The current notation to refer to an SBML element inside an annotation block is shown in Listing 4.4.

```
<listOfSpecies>
 <species metaid="metaid_0000042" id="Y" name="Calcium"
   compartment="intravesicular" initialConcentration="0.36">
  <annotation>
   <rdf:RDF>
    <rdf:Description rdf:about="#metaid_0000042">
     <bqbiol:is>
      <rdf:Bag>
       <rdf:li rdf:resource="urn:miriam:obo.chebi:CHEBI%3A22984"/>
      </rdf:Bag>
     </bqbiol:is>
    </rdf:Description>
   </rdf:RDF>
  </annotation>
 </species>
</listOfSpecies>
```

Listing 4.4: SBML element reference

The sample listing shows the annotation of an SBML `species` element. It is currently not possible to further annotate the `initialConcentration` attribute. One might, for example, want to specify a reference to the relevant literature that justifies the provided value, or to say *who* set the concentration value.

Contrary to that, the Annotation package allows to refer to XML attributes. While elements are addressed by `rdf:about="#_elementMetaID"`, an attribute is referred to using XPath [Clark and DeRose 1999]. XPath is a standard technology for referencing elements and attributes inside an XML document. It offers a well defined scheme, and a great number of tools exist for the evaluation of XPath expressions. Using the XPath concept in the `rdf:about` attribute enables the addressing of any local object. The general scheme is: `rdf:about="xpath:XPathToTheObject"`. Listing 4.5 shows an example.

```
<species metaid="metaid_0000042" id="Y" name="Intravesicular Calcium"
         compartment="intravesicular" initialConcentration="0.36">
 <annot:annotation>
  <rdf:RDF>
   <rdf:Description
    rdf:about="xpath://species[@id='Y']/@initialConcentration">
    <bqbiol:description>
     <rdf:li rdf:resource="urn:miriam:pubmed:12343565"/>
    </bqbiol:description>
   </rdf:Description>
  </rdf:RDF>
 </annot:annotation>
</species>
```

Listing 4.5: SBML Annotation package attribute reference

Here the initial concentration of species `Y` is annotated with a PubMed article reference, using the qualifier `bqbiol:Description`.

Within the Annotation package, XPath must not be used to address attributes and elements by their (ordering) number (e. g. `//species[7]/@initialConcentration`) as SBML *per se* does not give meaning to the element order. Instead, the abbreviated XPath syntax as specified in the W3C recommendation [Clark and DeRose 1999, Sec. 2.5] must be used to identify an XML element by its id and then refer to the particular attribute (e. g. `//species[@id="Y"]/@initialConcentration`).

4.2.5. Annotations about annotations

To provide a statement about an existing annotation, both must relate to each other (e. g. to state that "The annotation with ID `rdf:about="#_000002"` has been provided by a person called 'John Doe'."").

The current SBML annotation standard does not support the `rdf:ID` concept. It is therefore impossible to refer to an `rdf:Description` block. To enable annotations about annotations in the future, the Annotation package proposes the use of the RDF *reification* concept [Miller and Manola 2004, Sec. 4.3]. Reification allows to assign further statements to any statement carrying an `rdf:ID`. Subsequent statements refer to a statement by specifying the `rdf:ID` in the `rdf:about` attribute of the `rdf:Description` attribute. An example for an annotation about another annotation is given in Listing 4.6.

```
1  <species metaid="metaid_0000042" id="Y" name="Intravesicular Calcium"
2           compartment="intravesicular" initialConcentration="0.36">
3   <annot:annotation>
4    <rdf:RDF>
5     <rdf:Description rdf:about="xpath://species[@id='Y']/
          @initialConcentration">
6      <bqbiol:description  rdf:ID="statement1">
7       <rdf:li rdf:resource="urn:miriam:pubmed:12343565"/>
8      </bqbiol:description>
9     </rdf:Description>
10    <rdf:Description rdf:about="#statement1">
11     <dc:creator>John Smith</dc:creator>
12    </rdf:Description>
13   </rdf:RDF>
14  </annot:annotation>
15 </species>
```

Listing 4.6: SBML annotations about annotations

The example shows two statements: The first one annotates a species attribute (ll. 5-10). The second statement (ll. 11-13) then specifies the creator of the first statement, saying that John Smith provided the first annotation.

Sometimes it is necessary to group different annotations together and relate them to a single subject. The Annotation package proposes to reuse the RDF *blank node* concept. For example, to express that "Hexokinase 2 is modified by phosphoserine in position 158" [Waltemath et al. 2011d] one needs to make two statements: First, "Hexokinase2 is modified by phosphorerine". Secondly, "The modification is observed at position 158"[2]. Both statements together need to be applied to the particular species. Listing 4.7 shows the encoding.

[2]`bqbiol:position` is not yet an agreed updon qualifier.

```
<species id="x" metaid="meta_x" name="Hexokinase 2">
 <annot:annotation>
  <rdf:RDF>
   <rdf:Description rdf:about=" xpath://species[@id='meta_x']/">
    <bqbiol:modification rdf:nodeID="node1"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="node1">
    <bqbiol:modifier rdf:resource="urn:miriam:obo.psimod:MOD%3A00046"/>
    <bqbiol:position rdf:datatype="xsd:integer">158</bqbiol:position>
   </rdf:Description>
  </rdf:RDF>
 </annot:annotation>
</species>
```

Listing 4.7: SBML unary statements [Waltemath et al. 2011d]

4.2.6. Person's meta-information

SBML annotations currently are limited in specifying information on persons involved in the model building, publishing, curating and maintaining process (e. g. model creator or model curator). Only the `dc:creator` can be specified for a model. The Annotation package, however, proposes to use the `dc:creator` to provide meta-information about different persons, allowing it on both the model itself and any of the model sub-elements. The implicit semantics of such an annotations is that such annotations are inherited from parent nodes when a given node is not annotated. For example, if a model element is annotated with a `dc:creator` but none of its sub-elements are, it is assumed that all sub-elements have been created by the model creator [Waltemath et al. 2011d].

4.3. Summary

Results

This chapter introduced the first draft proposal of an enhanced annotation format for SBML. The proposal has been developed together with Allyson Lister (University of Newcastle) and Neil Swainston (University of Manchester). It has been furthermore discussed on the `annot-discuss` mailing list. The first version of the Annot package proposal has been released without fully having solved all identified issues of the current SBML annotation approach. In particular, the absence of model constituents cannot yet be encoded.

However, the current draft is an improvement to the existing SBML annotation scheme. The main achievements are:

1. full support of RDF containers which allows to be more specific with collections of annotations about an SBML element,
2. support of the RDF `ID` concept which enables annotations about annotations,
3. and thereby the further explanation of model constituents but also of the annotations provided about a constituent.

The SBML extension has been published as the *SBML Level 3 Package Proposal: Annotation* [Waltemath et al. 2011d].

Discussion and future prospects

The identified short-comings in the current SBML annotation standard have been discussed during the "Modeling biological systems: Bio-model similarities and differences based on semantic information" Workshop[3] in Rostock in February 2010. Consequently, the development of an SBML Level 3 package specifically addressing solutions for the annotation of SBML models and its constituents followed. A second meeting in May 2010 resulted in a first draft proposal for that package[4].

Application on πML As shown earlier in this work, the *current* SBML annotation scheme can be adopted in other representation formats. For example, CellML reuses the MIRIAM URNs as part of the SBML annotation scheme. While the Annotation package has been advocated as an SBML Level 3 extension, it is applicable to other representation formats, too. The annotation scheme is generic as it builds on the use of standard RDF and biomodels.net qualifiers. It can, for example, be applied with πML (Section 2.2.3). While the πML language definition does not *per se* support annotations, an extension can easily enable annotation support. To realize annotations in πML, the following minimal changes to the schema have to be made:

- Add RDF namespace to the language to support RDF annotations. Alternatively, import the SBML annotation package.
- Add a `<piml>` root element to the language that holds the original πML model and the annotations.
- Add an optional `<annotation>` element to the XML Schema. It must appear outside the `<pimodel>` element.

[3]`http://www.informatik.uni-rostock.de/~dk103/meetings/modelMeeting10`, last accessed 14 March 2011.

[4]`http://sbml.org/Events/Other_Events/Annotation_package_workshop_2010`, last accessed 14 March 2011.

- Add a mandatory `metaid` attribute to all language elements (i. e. the `pimodel` element and all occurrences of `definition`, `process`, `channel` elements). The attribute is needed to refer to particular model parts from the annotation section.

An `<rdf:Description>` block inside the `<annotation>` element must be created to provide meta-information about either the πML model (model-qualifier or biology-qualifier) or its constituents (biology-qualifier). The `rdf:about` attribute must contain a reference to an XML element ID or one of its attributes inside the `pimodel` element, or the `pimodel` element itself. Following the Annotation package proposal, the `rdf:Description` block must have a qualifier specified that defines the relation of the annotation to the referenced XML element. Listing 4.8 provides an example for the Annotation package-compliant annotation of a πML model constituent.

```
<pimodel metaid="metaid_000001">
  [..]
  <definitions>
    <!-- Euglena Prozess -->
    <definition name="euglenaSpecies" metaid="metaid_000005 ">
      <annot:annotation>
       <rdf:Description rdf:about="xpath://definition[@metaid=
        'metaid_000005']">
         <bqbiol:identity rdf:resource="urn:miriam:taxonomy:3038"/>
       </rdf:Description>
      </annot:annotation>
      [..]
    </definition>
    [..]
  </definitions>
  [..]
</pimodel>
```

Listing 4.8: Annotation of a πML process definition using the Annotation package proposal

Here, the process `definition` with `metaid="metaid_000005"` is annotated as being the *Euglena* species, following the definition given in the NCBI Taxonomy. A detailed example is given in Appendix A.3.2. In the same way, CellML model elements can be annotated (Appendix A.3.3).

Negations of annotations A remaining issue is the encoding of negative knowledge (i. e. encoding the fact that something is known to be untrue).

One way to implicitly encode negations is the creation of a closed `rdf:list` which explicitly names all true objects (URNs) for a given subject (model constituent) with regard to a particular predicate (qualifier). A list implies that all unlisted objects are excluded from the semantic description.

There is, however, a difference between listing all valid objects, and specifically excluding a number of objects. For example, the following three statements differ in their meaning:

- "species `#metaid_0000042` `bqbiol:is` (`A` and `B`)", open set of characteristics (e. g. using the `rdf:Bag` concept),
- "species `#metaid_0000042` `bqbiol:is` only (`A` and `B`)", closed set of characteristics using the `rdf:List` collection concept), and
- "species `#metaid_0000042` `bqbiol:is` not `(C)`".

The first statement lists a possible sub-set of characteristics that is not necessarily complete. The second one fully determines the characteristics and ensures that no further definitions of the subject exist. The third statement determines a sub-set of invalid characteristics, without saying anything about possible other characteristics.

Statement one is supported by the current SBML annotation standard, as well as by the Annotation package proposal (Listing 4.1).

Statement two is not supported by the current SBML annotation scheme as collection support is missing. The problem can be solved with the Annotation package as it allows to use `rdf:list` elements: If we can *list* the true objects for a given subject, we can infer that all other possible statements are not true and as such could be considered negative annotations. However, given the diversity and complexity of biological systems and considering the fact that many things are still unknown, this approach does not seem feasible.

None of the discussed annotation schemes supports statement three as they do not have negation mechanisms defined. However, these mechanisms are especially important as the finding of a negative fact often needs to be explicitly stated ("I found that species X is *not* phosphorylated."). One proposed solution is the extension of currently existing *qualifiers* by their negative counterparts. For example, the bio-qualifier `is` will be complemented by the bio-qualifier `isNot`. Listing 4.9 shows an example. The annotation states that reaction `metaid_905823` is a degradation process (`urn:miriam:obo.sbo:0000179`) but *is not* a transport reaction (`urn:miriam:obo.sbo:SBO:0000185`).

```
<listOfReactions>
 <reaction id="Reaction1" metaid="metaid_905823" name="degradation of
     LacI transcripts" reversible="false" sboTerm="SBO:0000179">
  <annot:annotation>
   <rdf:RDF>
    <rdf:Description rdf:about="#metaid_905823">
     <bqbiol:identity>
      <rdf:Bag>
```

```
        <rdf:li rdf:resource="urn:miriam:obo.sbo:0000179"/>
       </rdf:Bag>
      </bqbiol:identity>
     </rdf:Description>
     <rdf:Description rdf:about="#metaid_905823">
      <bqbiol:isNot>
       <rdf:Bag>
        <rdf:li rdf:resource="urn:miriam:obo.sbo:SBO:0000185"/>
       </rdf:Bag>
      </bqbiol:isNot>
     </rdf:Description>
    </rdf:RDF>
   </annot:annotation>
   <listOfReactants [..] />
   <kineticLaw [..] />
  </reaction>
</listOfReactions>
```

Listing 4.9: Use of negative qualifiers in the Annotation package

One remaining problem is how to explicitly state the *absence* of a constituent.

As none of the so far suggested options fully satisfied all needs, supporting negation statements might demand moving to OWL[5].

Semantics of RDF lists The Annotation package supports all types of RDF containers. The `rdf:List` defines a closed list of ordered references. However, the semantics behind that order are not clear, and they are not defined by the Annotation package as the `rdf:List` may be used in various different annotations (e g. in a list of modification dates, in a list of species descriptions, or in a list of publication references). The context determines the semantics of the listing (e. g. temporal order, quality, or relevance). A mechanism to specify the `rdf:List`'s semantics is not yet suggested.

Consistence of annotations Currently, the person or software working with a model is fully responsible for the annotations' consistence as the model code evolves. With two different annotation schemes (i. e. the SBML standard annotation and the Annotation package) inconsistencies may occur easily. For example, one software tool may not support the Annotation package and therefore add and update existing SBML standard annotations. These annotations might then contradict with the ones encoded in the Annotation package proposal.

[5]This suggestion was made by Nadia Anwar during the first COMBINE meeting, Edinburgh, October 2010, `http://sbml.org/Events/Forums/COMBINE_2010`, last accessed 14 March 2011.

Annotating experimental data Another idea for future work is to integrate experimental and model data for the retrieval approach. As the retrieval is based on meta-information, existing experimental data needs to be annotated in order to be comparable to the model and to the simulation result data. One concrete application where annotation of experimental data is needed is the Mosan tool [Unger 2010] that allows for the visualization of experimental and simulation runs. In order to find suitable bio-models for the result data sets, a matching between the entities used in Mosan and the components stored in *mDB* must be realized. Annotations – both in the old and the new annotation scheme – are a promising concept to achieve that goal. The approach is described in more detail in Section 6.6.

Converting the proposal into a package The main future goal is the acceptance of the Annotation package proposal as an SBML extension. Therefore, the package proposal as discussed during the Annotation package meeting[6] and published in [Waltemath et al. 2011d] has to be implemented by two software tools. Once the extended annotation scheme is used for the annotation of SBML models, evaluations will take place.

[6]`http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Annotations`, last accessed 19 October 2010.

5. Exchange and reuse of simulation experiments

An unplanned, hit-or-miss course of experimentation with a simulation model can often be frustrating, inefficient, and ultimately unhelpful.

(Kelton and Barton [2003])

One important aspect of meta-information related to bio-models is the information about *simulation experiments applicable to the model.* To benefit computationally from such information, it must be encoded in a standardized format.

This chapter introduces a guideline for the minimum information necessary to describe a simulation experiment, a format for the encoding of such simulation experiment descriptions, and an ontology for the classification and characterization of simulation algorithms. First, Section 5.1 discusses the drawbacks of the current situation. Section 5.2 then describes the minimum information guideline in detail (MIASE). A data model for the description of MIASE-compliant simulation experiments is the *Simulation Experiment Description Markup Language* (SED-ML, Section 5.4). SED-ML utilizes an ontology to refer to simulation algorithms. The ontology is called *Kinetic Simulation Algorithm Ontology* (KiSAO, Section 5.3). The chapter concludes with implementation details (Section 5.5) and a discussion (Section 5.6).

The MIASE guidelines Waltemath et al. [2011a] are considered by different areas in computational biology, including neuroscience, cell physiology, or drug discovery. A first version of SED-ML has been published [Waltemath et al. 2011b], and support is currently being implemented by different software tools[1].

5.1. Problem statement

By stating that "any scientific activity should be based on controlled and independently repeatable experiments", Pawlikowski et al. [2002] advocate to apply the

[1]SED-ML support is listed on `http://sed-ml.org/`, last accessed 21 March 2011.

scientific method to simulation, resulting in a situation where “many repetitions [of a non-sequential simulation] can eventually obtain the final results with acceptably small statistical errors”.

One goal of the modeling process is to understand a systems’ behavior by running simulations on it [Kelton and Barton 2003]. An important aspect when aiming at reusing a model is the existence of a valid, useful and documented computational experiment that can easily be performed on the model. The first prerequisite is successful retrieval of a relevant model description. The next step then is to simulate that model to obtain a desired output, often based on an existing simulation protocol [Waltemath et al. 2011a]. At the same time, a model with an associated, valid simulation experiment can be judged “more relevant” than a model that has no proof of correct working associated with it.

The state-of-the-art procedure of applying an experiment on a bio-model is to read the simulation description in the corresponding publication. This is an error-prone and time-intense process. It is often impossible to reconstruct the simulation setup from the information given in the literature. In many cases, the information does not comply with the final setup, it is wrong or out-dated. In addition, the textual simulation description in a particular (for many researchers foreign) language might be ambiguous and imprecise. In those cases, finding the correct information required for the simulation steps must rely on “educated guesses”. Sometimes, hints on the parametrization are given in the `<notes>` tag inside the model representation format (e. g. Listing A.1, ll. 61-66) but the description is non-standardized and informal. It cannot be automatically extracted.

At present, software does not provide means to retrieve experiments together with a bio-model. Providing a checklist of simulation-relevant information, however, will increase the efficiency of simulation experiment reuse, and thereby also enhance bio-model reuse. Experiment setups require a very detailed knowledge of the model and the modeled system, including important inputs to the model, degree of sensitivity to changes in the input, usage of random numbers, applicable simulation algorithms, post-processing of simulation results and output performance measures [Kelton and Barton 2003; Waltemath et al. 2011a]. The encoding of such information saves time and effort when repeating simulation experiments on models [Nickerson et al. 2008; Waltemath et al. 2011a]. If simulation experiments can be reused, they can also be exchanged among researchers, fostering co-operations and simplifying communication and discussion about models and their shown behavior.

The aforementioned co-existence of representation formats (e. g. SBML, CellML, and NeuroML) entails the existence of a great number of different simulation tools [Beard et al. 2009; Endler et al. 2009]. None of them supports more than a sub-

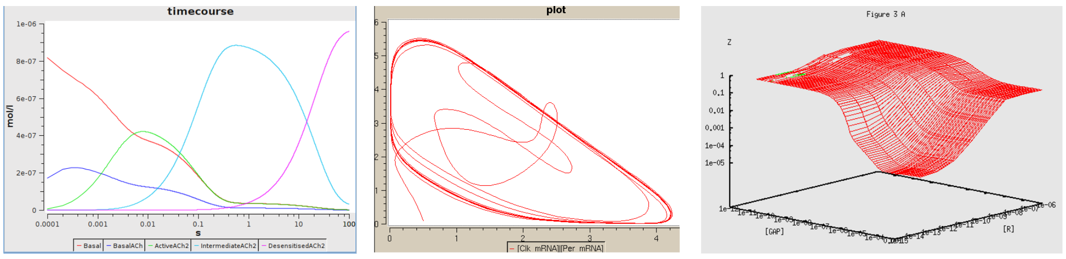


Figure 5.1.: Examples for complex simulation experiments. Extracted from [Le Novère 2009], first presented in [Le Novère 2008]. The left diagram presents the result of a time course simulation plotting different biological entities on a logarithmic x-axis. The original simulation results are given in the reference publication [Edelstein et al. 1996]. The middle diagram shows the result of a phase plane analysis of the biological entities `mRNA` and `Per_mRNA` as described in [Ueda et al. 2001]. The right diagram shows a more complex post processing with a three-dimensional visualization of the simulation result. The plot represents the simulation results described in the reference publication [Borneimer et al. 2004]. .

set of the existing standards. Moreover, many existing tools do not support the import and export of their internal simulation experiment description, but only of the model used in the simulation itself. This situation restricts a user to the particular simulation tool used in the reference description. Therefore, simulation experiments run on the models may remain inaccessibly unless the user decides to “learn” another software or to adapt the simulation experiment.

The description of a simulation experiment can be trivial (e. g. running a simple timecourse simulation) but usually it is complex; Figure 5.1 shows some examples. From the sole plot, it is impossible to infer the applied post-processing on the model. Also, potential changes in the initial model parametrization are not indicated. A simulation experiment description format enables those experiments’ reuse.

5.2. Minimum Information About a Simulation Experiment

The *Minimum Information About a Simulation Experiment* (MIASE, [Waltemath et al. 2011a]) guidelines are the outcome of investigations specifying the core information to be provided to users of existing models, so that they can perform defined simulations on those models. The success of MI guidelines in other fields of biology and CSB (e. g. checklists for microarray experiments (MIAME) or checklists for model annotation (MIRIAM, Section 3.1.3)) suggests that MI guidelines can also facilitate better simulation reuse in CSB. The MIASE guidelines have been developed together with scientists from various modeling and simulation groups in CSB (reflected by the list of authors on the corresponding publication [Waltemath et al. 2011a]). The guidelines and the format used to formalize the guidelines (Section 5.4) are considered by groups in the SBML, CellML and NeuroML communities.

Section 5.2.1 first describes the scope of the guidelines. Section 5.2.2 then explains the guidelines in detail. Finally, Section 5.2.3 provides an example for a MIASE-compliant experiment description.

5.2.1. Scope

The exchange and reuse of models in the field of computational biology have been advocated for several years now, and MIs such as MIRIAM [Le Novère et al. 2005] have become widely accepted. Once the models are retrieved, the next step is to test existing *simulation protocols* on them to obtain a desired output. Already the MIRIAM guidelines (*rule 6*) state that

> The model, when instantiated within a suitable simulation environment, must be able to reproduce all relevant results given in the reference description that can readily be simulated. [Le Novère et al. 2005]

How to enable reproducibility of the relevant results is not specified in MIRIAM as the procedure to fulfill this rule is not within its defined scope. That is why the MIASE effort set out to enable better reproducibility of simulation results in the field of CSB. The concept of reproducibility for the scope of this work is given in Definition 5.2.1.

Definition 5.2.1 (Reproducibility, adapted from [McNaught and Wilkinson 1997])**.** *The closeness of agreement between independent results obtained with the same method on identical test material but under different conditions (different operators, different apparatus, different laboratories and/or after different intervals of time).*

MIASE guidelines advocate for the *reproducibility* of a simulation experiment, opposed to the fact that MI guidelines are usually aiming at repeatability (i. e. redoing a simulation experiment with an identical experimental set-up) [Waltemath et al. 2011a].

The scope of MIASE is within calculations performed on bio-models, regarding the temporal and spatial evolution of a biological system. Examples are time series (i. e. the development of a system over time), parameter scans (i. e. performing a simulation for a range of values of certain parameters), but also sensitivity analysis, bifurcation studies and more. MIASE, being a reporting guideline [Taylor et al. 2008], concentrates on describing simulation experiments (i. e. how to report clearly and unambiguously what has been done, by describing the entities involved in the experiment). Contrarily, it does not provide information about the experiment's quality (i. e. statements on the correctness of experimental approaches, or on how an experiment should be performed). Consequently, MIASE does not require statements on the correctness of a simulation result nor the conditions under which a result is valid. That discussion is left to the potential users of the simulation experiment description. Also, MIASE does not demand the user to provide information on the models' structure, nor information on how to gain a valid model parametrization.

For the time being, MIASE is restricted to simulation descriptions of biological systems that could be (but are not necessarily) written with ordinary and partial differential equations [Waltemath et al. 2011a], that are mathematical models of biochemical and physiological systems. The reason for that restriction is the launch of the project in the CSB and physiome community. As stated in [Waltemath et al. 2011a], however,

> MIASE principles are of general applicability and should appeal to other communities. It can be expected that MIASE compliance will be directly applicable to a wider range of simulation experiments, being of help for other communities using similar approaches, such as computational neuroscience or ecological modeling. Furthermore MIASE could even be extended to cover other areas of mathematical modeling in life science, for instance process algebra.

5.2.2. Guidelines

For a simulation experiment description to be MIASE-compliant the following rules must hold (published in [Waltemath et al. 2011a]):

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,r 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 only 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 of how the model has to be run with the specified environment/platform in order to achieve the purpose of the experiment must be given.

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.

The rules specify the content of a MIASE-compliant simulation experiment description. Figure 5.2 visualizes the presented MIASE rules.

A simulation setup bases on one or more bio-models. As such, the first essential step is the *specification of all model(s)* involved in the simulation experiment

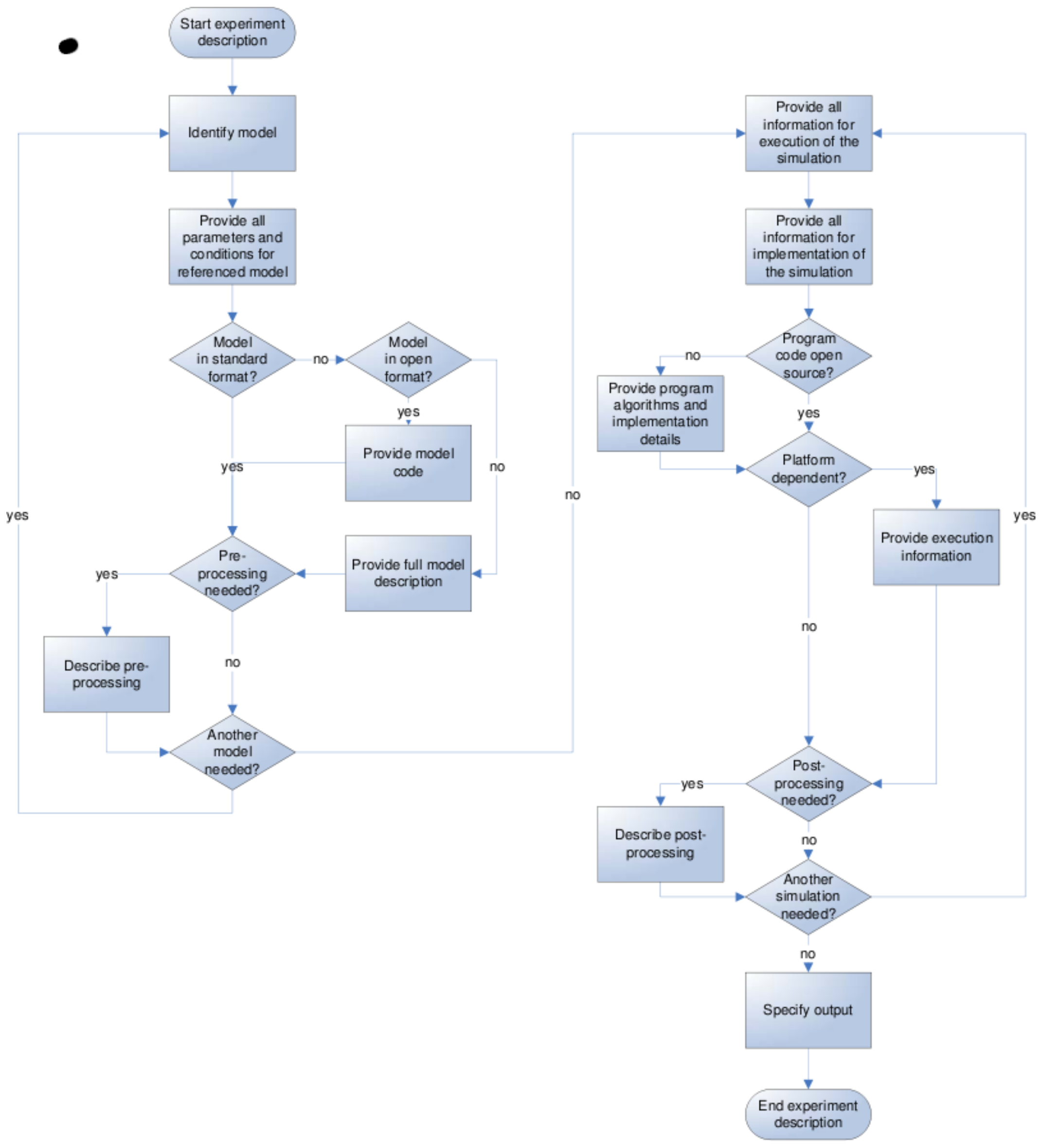

Start experiment description
Identify model
Provide all parameters and conditions for referenced model
Model in standard format?
no
Model in open format?
yes
Provide model code
no
yes
Pre-processing needed?
Provide full model description
yes
Describe pre-processing
no
Another model needed?
yes
no
Provide all information for execution of the simulation
Provide all information for implementation of the simulation
Program code open source?
no
Provide program algorithms and implementation details
yes
Platform dependent?
yes
Provide execution information
no
Post-processing needed?
yes
Describe post-processing
no
Another simulation needed?
yes
no
Specify output
End experiment description

Figure 5.2.: Flowchart representing the rules for a MIASE-compliant simulation (page 90). Rectangles represent processes, diamonds represent decision points [ISO 5807 1985].

(**rule 1**). Different ways of referencing a model from a simulation experiment description include joining the model with the experiment description, or referencing it through a link. The reference must, however, guarantee to be precise and unambiguous to ensure the model's long-term access (**rule 1A**). MIASE recommends the use of models in standard (representation) formats that comply with the MIRIAM guidelines. Examples are referenced models using BioModels Database URNs (`urn:miriam:biomodels.db:`*modelID*) or models from the CellML Model Repository (`http://models.cellml.org`). A `.zip` format delivering both models and the simulation description has been proposed recently by Frank Bergmann and can be used with the software framework *Systems Biology Workbench* (`http://sbw.sourceforge.net/`). It is recommended for models that are not publicly available (**rule 1C**). All models must be submitted together with all parameters, equations and conditions relevant for the simulation experiment (**rule 1B**). Furthermore, any preprocessing to be applied to a model before simulation must be precisely described in the experiment description (**rule 1D**). Examples are the update of a parameter or a mathematical function but also the range of a parameter in a parameter scan.

A MIASE-compliant simulation description contains all information necessary for the simulation itself (**rule 2**). The particular type(s) of simulation experiment to be run on the model must be specified (e. g. time course simulation, parameter scan, and bifurcation analysis). Together with the type, all necessary information for the simulation run must be submitted (e. g. the step size for a uniform time course simulation, or the particular simulation algorithm used to run the simulation). For multi-step simulations, the order of all simulation steps must be given. If the output of a simulation step must be processed before it is used in the follow-up simulation step, then the processing must be unambiguously described (**rule 2A**). The identification of the simulation algorithm and its variant used in the simulation experiment is particularly important as the numerical results often depend on the chosen implementation. MIASE recommends the use of controlled vocabulary to refer to a particular simulation algorithm. One proposal for an ontology of simulation algorithms is the KiSAO (Section 5.3). Sometimes the simulation run might depend on further parameters. Those have to be defined in the MIASE-compliant simulation description (e. g. the choice of random number generator for stochastic simulations or the meshing method used for discretisation in spatial simulations) (**rule 2B**). Same as for the availability of the models used in the experiment, also the simulation software must be available. If a closed-source software is used, all information necessary to reproduce the simulation must be provided (i. e. information to re-implement the software) (**rule 2C**). If a simulation run demands particular hardware setups or software libraries, those must be specified and explained in the simulation description (**rule 2D**).

Furthermore, MIASE requires information essential to obtain a particular simulation output (**rule 3**). The experiment description must contain all information about post-processing on the raw simulation output, if it has to be applied (**rule 3A**). Examples are the normalization of a data set or the output of data on a logarithmic scale. The particular type of output can lead to different insights of the system. It therefore has to be specified in a MIASE-compliant simulation description. Examples are the output as a 2D plot, a movie, or a simple data table. Plotting two variables on a time axis shows their change in concentration over a given range of time, while plotting them against one another leads to a phase portrait showing the dependency of both variables.

5.2.3. Example

The following simulation description of the LeLoup and Goldbeter model [Leloup and Goldbeter 1999][2] shows the application of the MIASE guidelines. The aim is to run a timecourse simulation of the LeLoup model reproducing the chaotic oscillation of the tim mRNA complex as shown in Figure 3B in the reference publication. A MIASE-compliant description of that experiment is:

1. Use the model `urn:miriam:biomodels.db:BIOMD0000000021` for the simulation experiment. (**rule 1A+1B**)
2. Update the model parameters. (**rule 1A**)
   - Update the initial value of the parameter `V_mT` to `0.28`. (**rule 1D**)
   - Update the initial value of the parameter `V_dT` to `4.8`. (**rule 1D**)
3. Run a timecourse simulation on the model. Use the `CVODE solver` described in `Cohen S, Hindmarsh C:` *`Cvode, A Stiff/nonstiff Ode Solver`In: Computers in Physics, Vol. 10 (2), pages 138-143 (1996)* for that simulation. (**rule 2A**).
4. Run the timecourse simulation of the model for 1000 time units, taking measurements after every single time unit. (**rule 2A**)
5. After simulation, create a 2-dimensional plot with one curve showing time on the x-axis and the values of `Mt` on the y-axis. (**rule 3B**)

[2] "Chaos and birhythmicity in a model for circadian oscillations of the PER and TIM proteins in drosophila", Appendix A.2.1.

The first item in the enumeration defines the model used in the simulation. In this example, the referenced model (`urn:miriam:biomodels.db:BIOMD0000000021`) is available from BioModels Database. Resolving the URN leads to its SBML encoding. SBML provides the model with all necessary parameters and the corresponding initial values, as well as the reactions taking place.

The second item in the enumeration describes the necessary parameter updates before simulation. The original model is changed by applying two updated parameter values: a new value for the parameter `V_mT` and a new value for the parameter `V_dT`.

The third and fourth item in the enumeration refer to the simulation setup itself. The model should be simulated in a uniform timecourse simulation over 1000 time units. The simulation algorithm to be used is the `CVODE` solver; the algorithm is specified through an informal reference to literature.

The last item in the enumeration then defines the output of the simulation as being a 2-dimensional plot. The values of `Mt` are shown in the plot.

`http://libsedml.svn.sourceforge.net/` provides an extended version of the example which compares the `Mt` value in the original parametrization with the one using an updated parametrization.

5.3. Kinetic Simulation Algorithm Ontology

One important part of a simulation experiment is the simulation algorithm used to solve the system. Simulation results may differ depending on the chosen algorithm [Waltemath et al. 2011a]. The sole reference of a simulation algorithm through its name in form of a string is error prone and unambiguous. Firstly, typing mistakes or language differences may make the identification of the intended algorithm difficult. Secondly, many algorithms exist with more than one name, having synonyms or various abbreviations that are commonly used [Waltemath et al. 2011b]. MIASE demands reference to an unambiguous source of simulation algorithms for each simulation setup specified (**rule 2A**). An ontology of existing simulation algorithms allows to provide that information.

Ontologies are already used in the context of M&S. Fishwick and Miller [2004], for example, show a small ontology for Petri Net types. Another example is the *Discrete-Event-Modeling Ontology* (DeMO, [Miller et al. 2004]), an ontology for discrete-event modeling and simulation. It covers several types of discrete-event models, and is built of four different hierarchies (event-, state-, activity- and process oriented models) [Fishwick and Miller 2004]. The developers aimed at capturing knowledge about the DE modeling domain. They identified, from their point of view, all relevant concepts and provided (where possible) machine-interpretable formal specifications for these concepts [Miller et al. 2004]. However, DeMO is very restricted in its scope as it does not address much of the modeling and simulation domain – for example, continuous models, statistical modeling, output analysis, random variates [Silver et al. 2007] are not covered. An ontology for simulation algorithms has so far not been developed.

The *Kinetic Simulation Algorithm Ontology* [Courtot et al. 2011] (KiSAO) is an ontology for the classification and organization of simulation algorithms mainly used for kinetic simulation experiments. KiSAO development is work at an early stage; current versions are available in OBO[3] and OWL format. The ontology is maintained as a sourceforge project[4]. It is part of the biomodels.net effort[5] and it is registered at the bioportal[6] which also provides a visual KiSAO browser. KiSAO consists of 61 classes, covering 53 algorithms and their variants.

Classification KiSAO has four main branches (i. e. categories) with several sub-branches (Figure 5.3). The branches implement general concepts such as *algorithm using adaptive timesteps* ($KISAO$: 0000041) and more specific ones such as *Gillespie-like approximate simulation method* ($KISAO$: 0000001). Each of the concept classes characterizes the contained simulation algorithms. As an example, Figure 5.3 shows the classification of Gillespie's first reaction method: It is a *stochastic* algorithm with *adaptive time steps* that uses *discrete* variables and disregards spatial information (*algorithm using non-spatial description*). From the ontology, one can infer that Gillespie's first reaction method is a special case of the more general *optimized reaction method.* Both belong to the class of *Gillespie-like exact stochastic simulation method.*

Characterization Each algorithm class has a set of attributes attached to it which characterizes the encoded algorithm. Properties include a definition that holds a link to the reference publication, possible synonyms for the algorithm names, and a label. For the example of Gillespie's first reaction method, the corresponding entry in the OBO version of the ontology is shown in Listing 5.1.

Use cases Information on the simulation algorithm can help in finding relevant models, for example, when searching for models solvable in a particular simulation environment. KiSAO is used in SED-ML to reference a particular simulation algorithm from a simulation setup (Section 5.4.2). Furthermore, the `kisaoID` is stored

[3]`http://rest.bioontology.org/bioportal/ontologies/download/40844`, last accessed 14 March 2011.

[4]`http://sourceforge.net/projects/kisao/`, last accessed 10 August 2010.

[5]`http://biomodels.net/kisao`, last accessed 15 December 2010.

[6]`http://www.bioportal.org/bioontology`, last accessed 15 December 2010.

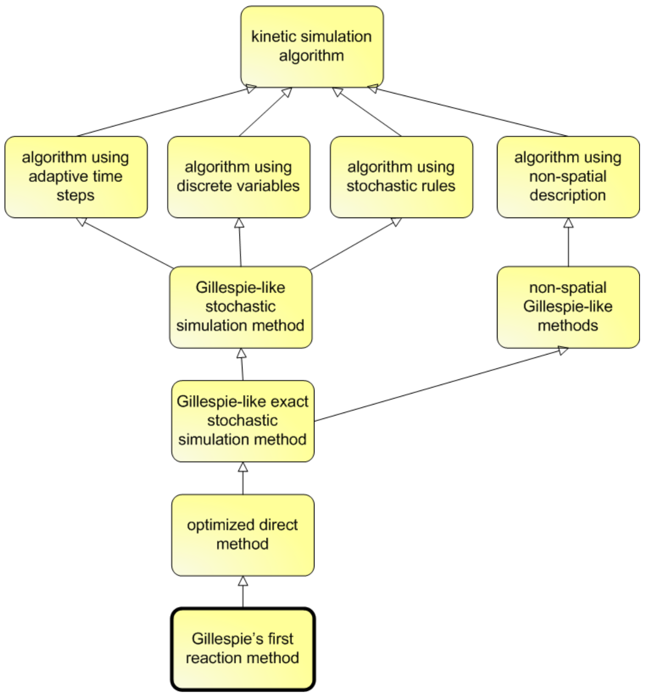

kinetic simulation algorithm
algorithm using adaptive time steps
algorithm using discrete variables
algorithm using stochastic rules
algorithm using non-spatial description
Gillespie-like stochastic simulation method
non-spatial Gillespie-like methods
Gillespie-like exact stochastic simulation method
optimized direct method
Gillespie's first reaction method

Figure 5.3.: Example for the classification of the "Gillespie's first reaction method" (KISAO:0000015).

```
[Term]
id: KISAO:0000015
name: Gillespie's first reaction method
def: "Stochastic simulation algorithm using the next-reaction density
     function, [..]. Gillespie DT. A General Method for Numerically
     Simulating the Stochastic Time Evolution of Coupled Chemical
     Reactions. J.CompPhysics, Volume 2 , pages 403-434 (1976)."
is_a: KISAO:0000034 ! optimized direct method
```

Listing 5.1: The characterization of Gillespie's first reaction method

in the simulation database *sDB* (Section 5.4.3); it forms one characteristic for the model retrieval and ranking approach (Chapter 7).

5.4. Simulation Experiment Description Markup Language

Existing guidelines need to be converted into normative standards [Sherman 2009] (citing [Burgoon 2006]). The MIASE guidelines have been implemented in an XML Schema [W3C 2004] representing the *Simulation Experiment Description Markup Language* (SED-ML) [Köhn and Le Novère 2008]. The schema has been defined based on a UML model which has been automatically transformed into a preliminary XML Schema using *Magic Draw UML 16.0.* The result has then been manually adjusted. Appendix A.4 shows both, the UML diagram and the full XML Schema.

The first official version of SED-ML is *SED-ML Level 1 Version 1 Release Candidate 1* (in the following referred to as *SED-ML*). It has been developed in co-operation with several modeling and simulation groups in CSB. SED-ML meetings started off as a regular session during SBML-related meetings; the first meeting discussing SED-ML was the *12th SBML Forum Meeting* in 2007. Later on, SED-ML discussions moved to combined efforts meetings such as the *Super-hackathon "standards and ontologies for Systems Biology"*[7] in 2008, the *combined CellML-SBGN-SBO-BioPAX-MIASE 2009 workshop*[8], and the *COmputational Modeling in BIology NEtwork*[9] (COMBINE) meeting in 2010.

Section 5.4.1 first introduces the SED-ML URN concept which is one basic reference mechanism in SED-ML. Section 5.4.2 provides an overview of the SED-ML language. The full language specification of the first SED-ML Release Candidate is available online from `http://sourceforge.net/projects/sed-ml/` and described in [Waltemath et al. 2011b]. Section 5.4.3 briefly discusses possibilities for the extraction and storage of SED-ML information in a database for later use in the retrieval process.

5.4.1. URI scheme

The URI Scheme is applied at several points in SED-ML: to refer to *model encodings* (i. e. publicly available model representation files), to specify the *language* of the referenced model, to address *implicit model variables*, and to *annotate* SED-ML elements. A standardized URI Scheme ensures long-time availability of a resource; that resource can unambiguously be identified.

[7]`http://sbgn.org/Events/SBGN-3.5`, last accessed 14 March 2011.
[8]`http://www.cellml.org/workshop/workshop2009/`, last accessed 14 March 2011.
[9]`http://sbml.org/Events/Forums/COMBINE_2010`, last accessed 14 March 2011.

Model references The preferred way to refer to a bio-model is the *MIRIAM reference standard* (Section 3.2.1). `urn:miriam:biomodels.db:BIOMD0000000048`, for example, points to the model with ID `BIOMD0000000048` in BioModels Database. `miriam` indicates that the URN is a standard MIRIAM URN. `biomodels.db` indicates that the resource is BioModels Database. SED-ML does not specify how to resolve the URNs. However, web services offered by MIRIAM Resources can be used to do so [Laibe and Le Novère 2007]. For the above example, the resolved URL may look like:

- http://biomodels.caltech.edu/BIOMD0000000048 or
- http://www.ebi.ac.uk/biomodels-main/BIOMD0000000048

depending on the physical location of the resource chosen to resolve the URN.

Language references A model's encoding is specified by a *SED-ML standard URN* (Appendix A.4.1) with the structure `urn:sed-ml:language:`*language*. SED-ML allows to generally specify the model representation format as `XML`, if no standardized representation format has been used to encode the model (`urn:sed-ml:language:xml`). But one can be as specific as defining a model as "SBML Level 2, Version 2" (`urn:sed-ml:language:sbml.level-2.version-2`)[10]. The language URNs can be resolved into URLs pointing to the language specification[11].

Symbol URNs Some variables used in a simulation experiment are not explicitly defined in the model, but may be implicitly contained in it. For example, to plot a variables behavior over time, the variable is defined in an SBML model; however, time is not explicitly defined in the model.

Implicit variables (so-called *symbols*) provide SED-ML with a standardized scheme to overcome that shortcoming. Symbols are defined externally in the SED-ML namespace, following the idea of MIRIAM URNs. The corresponding SED-ML URN scheme is `urn:sedml: symbol:`*implicit variable*. To refer from a SED-ML file to the definition of time, for example, the URN is `urn:sedml:symbol:time`. The list of predefined symbols is available from `http://biomodels.net/sed-ml`.

SED-ML URNs only provide a standardized location for implicit variables. From each such URN a mapping of SED-ML symbols on possibly existing concepts in the corresponding representation formats is provided.

[10]Language levels and versions are separated by a "`.`".

[11]For example, the resolved URL for the just given SBML version is `http://www.sbml.org/specifications/sbml-level-2/version-2/revision-1/sbml-level-2-version-2-rev1.pdf`

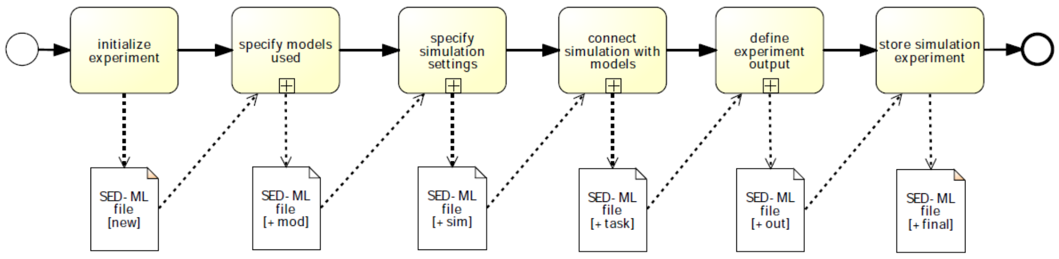


Figure 5.4.: Overview: The generic processes for the definition of a SED-ML file.

5.4.2. Language elements

I will explain the SED-ML data model using a workflow representation. It is described in the *Business Process Modeling Notation* (BPMN, [White 2004]). Images were created using the *Oryx workflow designer* [Decker et al. 2008] and its successor *Signavio Oryx V 4.1.5* (`http://academic.signavio.com`). Examples are partially taken from [Waltemath et al. 2011b]. Figure 5.4 gives an overview of the SED-ML workflow processes. All steps will be described in more detail and the most important concepts (instantiated as XML elements and attributes) will be discussed.

In a very generic sense, a SED-ML file is created by

1. defining all models participating in the experiment,
2. defining all simulation setups that will be used in the experiment,
3. connecting a model and a simulation setup at a time, and
4. defining the desired output.

To exemplify the use of SED-ML, a simulation experiment description of the "Chaos and birhythmicity in a model for circadian oscillations of the PER and TIM proteins in drosophila" model by LeLoup and Goldbeter [Leloup and Goldbeter 1999] is used. A (textual) MIASE-compliant description of the same experiment has already been given in Section 5.2.3.

Model specification

A simulation experiment can incorporate zero to many models which have to be unambiguously referenced and described (Figure 5.5).

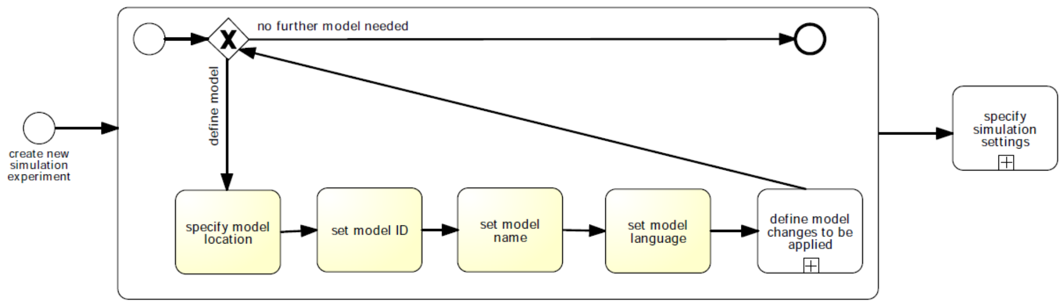


Figure 5.5.: Model specification process for the definition of a SED-ML file.

Model references The SED-ML approach uses the `source` attribute to point to any model file and the *model reference scheme* described in the URI scheme (Section 5.4.1).

Preferably, the model resource is defined by an unambiguous, consistent URI pointing to a single model in a particular version and encoded in an XML file (e. g. `source="urn:miriam:biomodels.db:BIOMD0000000021"`). Any resource registered at *MIRIAM resources*[12] can be used for model references. If the model is not yet published in a model repository registered with MIRIAM resources, then the model might be referenced using a `URL`[13]. However, long-term availability cannot be guaranteed for the references as URLs are not necessarily consistent. If the model code is not yet published, and SED-ML shall be used, then a reference to a local copy of the model code might be used (e. g. `source="./models/mymodel.xml"`). Exchange of such simulation experiment descriptions is limited though.

If a change is applied on a model, the `source` attribute references the original model by its SED-ML `modelID`.

Model encoding language Encoding the language for each referenced model is a helpful tool for simulation environments to quickly decide on whether or not they support a SED-ML description file (i. e. can load the model code). Again, SED-ML recommends the use of standardized URIs to refer to a particular representation format or model encoding (Section 5.4.1).

Listing 5.2 shows the generic XML `listOfModels` element containing two `model` element definitions.

[12]`http://www.ebi.ac.uk/miriam.main`, last accessed 14 March 2011.

[13]An example for a URL is `source="http://models.cellml.org/workspace/leloup_goldbeter_1998/@@rawfile/f14389c9271d0fe25e56c3cfc8d1a25d23c129af/leloup_1998b.cellml"`.

```
<listOfModels>
 <model id="model1" name="model name" language="URN to the language
     specification"
  source="URN to the corresponding XML file" />
 <model id="model2" name="perturbed version of model1"
  language="URN to the language specification" source="model1" >
  <listOfChanges>
   [description of perturbations]
  </listOfChanges>
 </model>
</listOfModels>
```

Listing 5.2: Definition of a set of models in SED-ML (general)

Listing 5.3 shows the definition of two models for the sample simulation. One originates from a public repository (`urn:miriam:biomodels.db:BIOMD0000000021`), and the second one is a perturbed version of the first model with applied changes.

```
<listOfModels>
 <model id="model1" language="urn:sedml:language:sbml.level-2.version-1"
  source="urn:miriam:biomodels.db:BIOMD0000000021" />
 <model id="model2" language="urn:sedml:language:sbml" source="model1">
  [..]
 </model>
</listOfModels>
```

Listing 5.3: Definition of a set of models in SED-ML (example)

The listing also shows the use of SED-ML language URNs: `model1` is characterised as an `SBML Level 2, Version 1` compliant model while `model2` is more generally defined as an `SBML` model. One possible reason can be that SBML Level 2 Version 1 compliance cannot be assured for the modified model version.

Model perturbations

Necessary changes on a model *before* simulation are defined in a `listOfChanges`, an optional sub-element of `model` (Figure 5.6).

Model code updates SED-ML distinguishes changes on the attribute values of a model's XML representation (`changeAttribute`), changes in the model's XML representation (`addXML, changeXML, removeXML`), and changes based on mathematical calculations (`computeChange`).

The first type updates the value of an XML attribute (e. g. changing a parameter value in the model, or updating an initial condition in SBML). The second type of change allows to exchange any XML code inside the model by another valid piece of XML (e. g. exchanging a compartment definition by another, or changing a rate law definition in SBML). The third type of change allows to change the definition of a

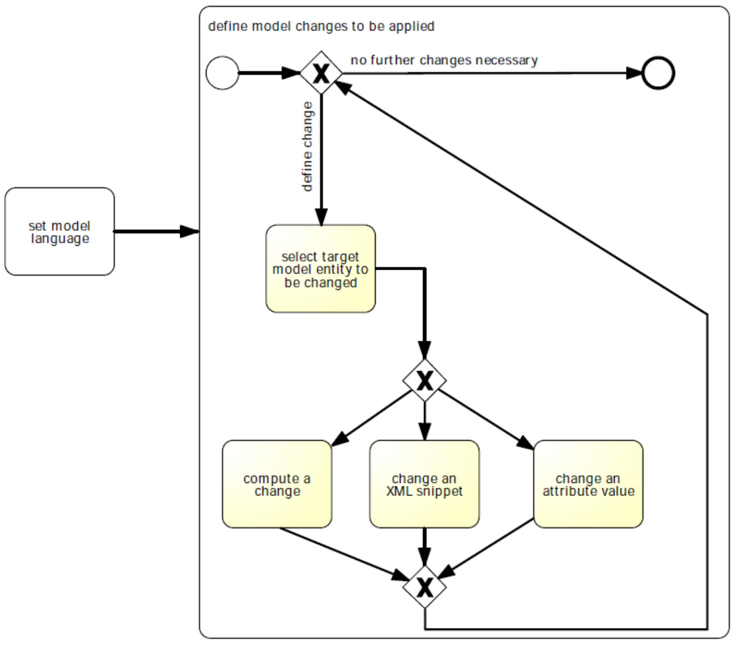

set model language
define model changes to be applied
no further changes necessary
define change
select target model entity to be changed
compute a change
change an XML snippet
change an attribute value

Figure 5.6.: Perturbation specification process for the definition of model perturbations in a SED-ML file.

mathematical function to calculate a new value for a model parameter or variable. For example, instead of giving a fixed value to a parameter in an SBML file, the value could be calculated based on the concentration of another model parameter. A mathematical expression may consider model variables and additional parameters for the calculations. Therefore, the `listOfVariables` and `listOfParameters` elements are introduced as sub-elements to the `computeChange` element.

Updates in the mathematics The SED-ML language reuses *MathML 2.0* [Poppelier et al. 2003] concepts to define both, pre-processing on the model before simulation and post-processing on the simulation results after simulation. Is is an international standard for encoding mathematical expressions in XML. It is also used by other standardization formats (e. g. SBML and CellML) to express mathematical constructs. As MathML is a very complex language, SED-ML uses a restricted subset of MathML 2.0 as defined in [Waltemath et al. 2011b, Sec. 2.1].

Code references Each type of change needs to address the particular piece of XML in the model encoding file. SED-ML uses XPath [Clark and DeRose 1999] to address nodes in the XML tree. XPath offers an unambiguous way of identifying an XML element or attribute in an XML file. In a SED-ML file, the `target` attribute stores the XPath expression.

The specification of all three types of changes is exemplified in Listing 5.4.

```
<model>
 <listOfChanges>
  <changeAttribute target="any attribute in the XML (XPath)"
   newValue="new value" />
  <changeXML target="any attribute or element in the XML"
   newXML="valid piece of XML declaration" />
  <computeChange target="any attribute or element in the XML (XPath)">
   <listOfVariables>
    [variables to be used for the calculation]
   </listOfVariables>
   <listOfParameters>
    [parameters to be used for the calculation]
   </listOfParameters>
   <math>
    [any expression compliant with the SED-ML MathML subset]
   </math>
  </computeChange>
 </listOfChanges>
</model>
```

Listing 5.4: Definition of model perturbations (generic)

Listing 5.5 defines a value change on the original LeLoup model. The parametrization of `model2` is specified based on `model1` and by updating the values of the two parameters `V_mT` (l. 7) and `V_dT` (l. 8).

```
1  <listOfModels>
2   <model id="model1" language="urn:sedml:language:sbml.level-2.version-1"
3    source="urn:miriam:biomodels.db:BIOMD0000000021" />
4   <model id="model2" language="urn:sedml:language:sbml.level-2.version-1"
5    source="model1">
6    <listOfChanges>
7     <changeAttribute target="/sbml:sbml/sbml:model/sbml:listOfParameters/
         sbml:parameter[@id='V_mT']/@value" newValue="0.28" />
8     <changeAttribute target="/sbml:sbml/sbml:model/sbml:listOfParameters/
         sbml:parameter[@id='V_dT']/@value" newValue="4.8" />
9    </listOfChanges>
10  </model>
11 </listOfModels>
```

Listing 5.5: Definition of a model perturbation (example)

Simulation setups

Similarly to the model definitions, a number of simulation setups can be defined (Figure 5.7). Simulation setups include the experiment type, but also the particular simulation algorithm, and the simulation settings.

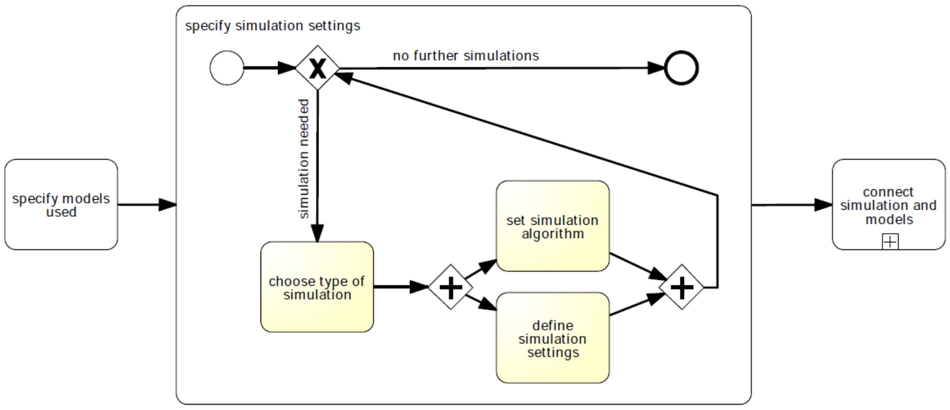

specify simulation settings
no further simulations
simulation needed
specify models used
choose type of simulation
set simulation algorithm
define simulation settings
connect simulation and models

Figure 5.7.: Simulation setup specification process for the definition of a SED-ML file.

Simulation types The first essential information is the type of simulation experiment. Examples are time course simulations, parameter scans, or steady state analyses. The SED-ML schema supports the encoding of uniform timecourse simulations (`uniformTimeCourse`). The following simulation settings are specified: the initial time (`initialTime`), the start time for the output (`outputStartTime`), and the end time for the output (`outputEndTime`) of the simulation. As a uniform timecourse simulation plots a curve at different yet uniform time points, either the step size or the number of points has to be provided. SED-ML stores the number of measurement points within the given time interval (`numerOfPoints`).

Simulation algorithms The last step in describing a simulation setup is the specification of the simulation algorithm. One algorithm must be defined for each simulation setup. SED-ML demands reference to controlled vocabulary; the use of *KiSAO* (Section 5.3) is recommended.

Listing 5.6 shows the general setup of a simulation.

```
<listOfSimulations>
 <uniformTimeCourse id="simulation setup ID" name="some simulation"
  initialTime="time" outputStartTime="start time" outputEndTime="end time
      "
  numberOfPoints="number of points">
  <algorithm kisaoID="KISAO:id" />
 </uniformTimeCourse>
 [FURTHER SIMULATION EXPERIMENT DEFINITIONS FOLLOWING]
</listOfSimulations>
```

Listing 5.6: Definition of simulation setups in SED-ML (generic).

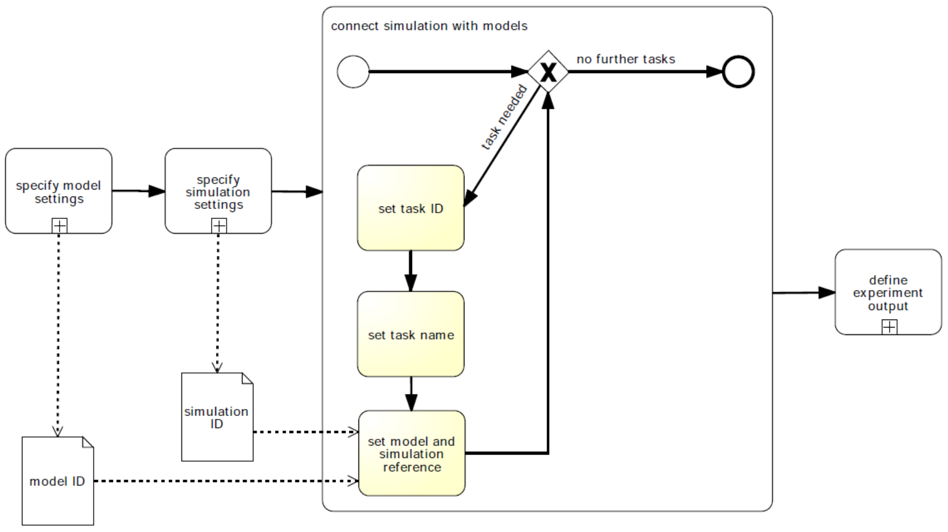


Figure 5.8.: Task definition process in a SED-ML file.

Listing 5.7 defines two particular time course simulations with uniform ranges, one using a deterministic algorithm (CVODE, $KISAO:0000019$) and the other one a stochastic algorithm (Gillespie Direct method, $KISAO:0000029$).

```
<listOfSimulations>
 <uniformTimeCourse id="s1" name="deterministic simulation"
  initialTime="0" outputStartTime="50" outputEndTime="500" numberOfPoints
      ="100">
  <algorithm kisaoID="KISAO:0000019" />
 </uniformTimeCourse>
 <uniformTimeCourse id="s2" name="stochastic simulation"
  initialTime="0" outputStartTime="50" outputEndTime="500" numberOfPoints
      ="100">
  <algorithm kisaoID="KISAO:0000029" />
 </uniformTimeCourse>
</listOfSimulations>
```

Listing 5.7: Definition of a simulation setup (example).

Assigning a simulation to a model

SED-ML constructs a simulation experiment from several different experiment setups. The modular definition of model and simulation setups allows SED-ML to execute elements of the model and simulation pools in different combinations. A pair of model and simulation setup is called a *task* (Figure 5.8). Each task has a mandatory `id`. It then consists of a model reference, pointing to one of the models in the model pool (`listOfModels`), and a simulation reference, pointing to one of the simulation setups in the simulation pool (`listOfSimulations`). Within a single task, only one model and one simulation can be grouped together. However, the definition of a set of tasks is possible. Both, models and simulations might be assigned to more than one tasks. Listing 5.8 shows the definition of a task in SED-ML.

```
<listOfTasks>
 <task id="task ID" name="task name" modelReference="reference to modelID
     "
  simulationReference="reference to simulationID" />
 [FURTHER TASK DEFINITIONS FOLLOWING]
</listOfTasks>
```

Listing 5.8: Definition of a task (generic)

Listing 5.9 shows the definition of two tasks, both using the same simulation setup, but different versions of a model. The first task runs simulation `s1` on model `model1`; the second task applies simulation `s1` to model `model2`.

```
<listOfTasks>
 <task id="t1" name="deterministic simulation of the original Leloup
     model"
  modelReference="model1" simulationReference="s1" />
 <task id="t2" name="deterministic simulation of the perturbed Leloup
     model"
  modelReference="model2" simulationReference="s1" />
</listOfTasks>
```

Listing 5.9: Definition of a task (example)

Post-processig simulation data

The simulation output specification is part of SED-ML. As a prerequisite, the necessary output data needs to be defined (Figure 5.9). The `dataGenerator` element encodes all relevant information to prepare the raw simulation data for the output.

If a data generator uses the raw output of the simulation, then it contains a simple reference to the particular piece of XML (e. g. to a species in SBML). However, in many cases, the raw result of the simulation is not sufficient to represent a phenomenon or to analyse the data. The post-processing steps have to be applied to the simulation result (e. g. normalization of data or calculation of mean values). SED-ML encodes post-processing through mathematical functions encoded in MathML. Each data generator contains the post-processing information to be applied to a data source. The data source does not necessarily have to come from inside the model. It is also possible to define "external" parameters using the URI concept (Section 5.4.1), for example to provide a definition of `time`. A whole pool of data generators can be created and later be reused in the output specification. List-

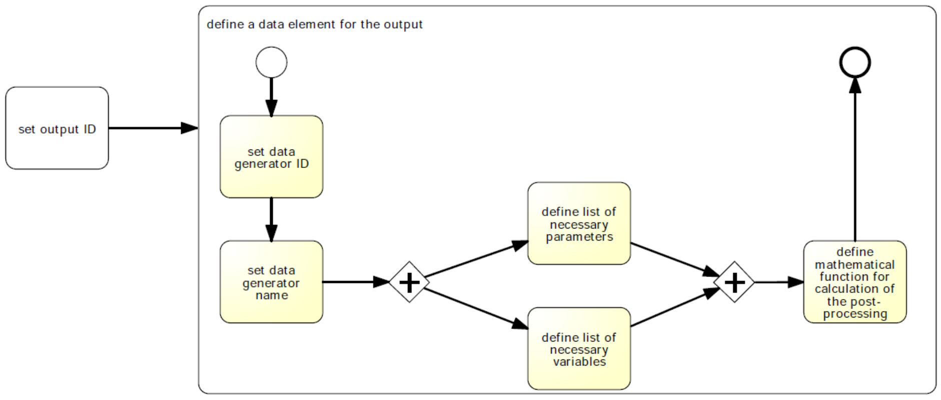

define a data element for the output
set output ID
set data generator ID
set data generator name
define list of necessary parameters
define list of necessary variables
define mathematical function for calculation of the post-processing

Figure 5.9.: DataGenerator definition in a SED-ML file.

ing 5.10 shows the generic structure of a data generator. In a SED-ML file, either the `target` or the `symbol` may be present; both must not occur at the same time.

```
<listOfDataGenerators>
 <dataGenerator id="data generator id" name="name"/>
  <listOfVariables>
   <variable id="variable id" name="some internal variable name"
    taskReference="task that contains the model in which the variable
        occurs"
    target="XPath to attribute or element in the model representing the
        variable"
    symbol="implicit variable" />
  </listOfVariables>
  <math>
   [mathML expression]
  </math>
  <listOfParameters>
   [PARAMETER DEFINITIONS]
  </listOfParameters>
 </dataGenerator>
 [FURTHER DATA GENERATOR DEFINITIONS]
</listOfDataGenerators>
```

Listing 5.10: Definition of dataGenerators (generic)

Listing 5.11 shows an example for the definition of two data generators, one referencing a model parameter (`d2`), the other one defining time (`d1`). Time is an implicit variable.

```
<listOfDataGenerators>
 <dataGenerator id="d1" name="time"/>
  <listOfVariables>
   <variable id="v1" taskReference="t1" symbol="urn:sed-ml:symbol:time"/>
  </listOfVariables>
  <listOfParameters />
```

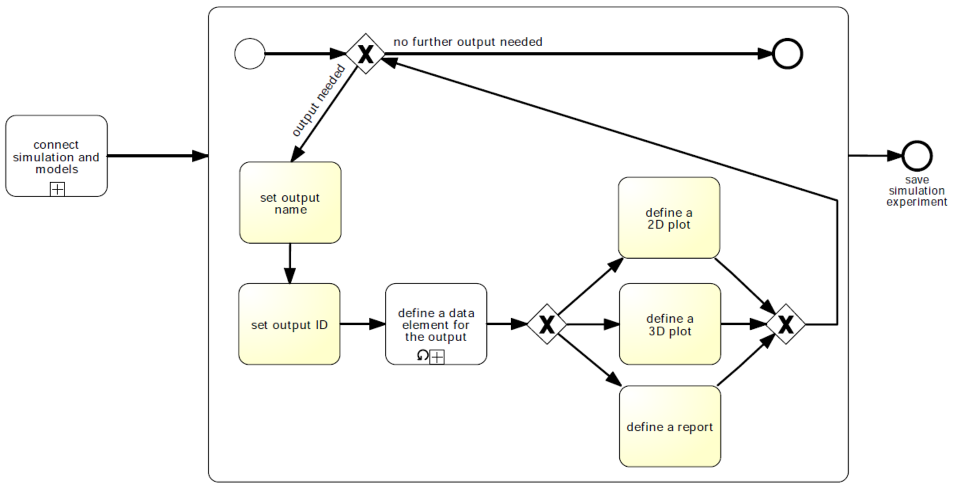


Figure 5.10.: Output definition process in a SED-ML file.

```
    <math>
     <ci> v1 </ci>
    </math>
  </dataGenerator>
  <dataGenerator id="d2" name="V_T" >
    <listOfVariables>
     <variable id="v2" name="V_T" taskReference="t1" target="/sbml:sbml/
         sbml:model/sbml:listOfParameters/sbml:parameter[@id='V_mT']" />
    </listOfVariables>
    <listOfParameters />
    <math>
     <ci> v2 </ci>
    </math>
  </dataGenerator>
</listOfDataGenerators>
```

Listing 5.11: Definition of dataGenerators (example)

Output specification

The output is specified based on the existing data generators (Figure 5.10); the different data sources (e. g. `d1` and `d2`) are mapped on the output parameters (e. g. axes of a plot or columns of a table). Every output definition is characterized by a certain output type. SED-ML supports *2D plot*, *3D plot* and *tables.* Depending on the output type, the necessary data sources are applied to the output components. For example, a two-dimensional plot requires the definition of both, an x-axis and a y-axis. One data source is assigned to each axis by linking to a particular data generator. Listing 5.12 shows the generic definition of outputs.

```
<listOfOutputs>
 <plot2D id="plot id" name="name for the plot" />
  <listOfCurves>
   <curve id="c1" logX="true/false" logY="true/false"
    xDataReference="data generator to be plotted on x-axis"
    yDataReference="data generator to be plotted on y-axis" />
   [FURTHER CURVE DEFINITIONS]
  </listOfCurves>
 </plot2D>
 <plot3D>
  <listOfSurfaces>
   <surface id="sf1" logX="true/false" logY="true/false"
    logZ="true/false"
    xDataReference="data generator to be plotted on x-axis"
    yDataReference="data generator to be plotted on y-axis"
    zDataReference="data generator to be plotted on z-axis" />
   [FURTHER SURFACE DEFINITIONS]
  </listOfSurfaces>
 </plot3D>
 <report>
  <listOfDataSets>
   <dataSet id="ds1" dataReference="data generator for report"
    label="label of data set" />
   [FURTHER DATA SET DEFINITIONS]
  </listOfDataSets>
 </report>
</listOfOutputs>
```

Listing 5.12: Definition of dataGenerators (generic)

An example for an output definition is given in Listing 5.13. It shows the specification of a two-dimensional plot. Time is plotted on the x-axis (referring to $d1$ in Listing 5.11). The V_T parameter value of both the original (referring to $d2$ in Listing 5.11) and perturbed LeLoup model ($d3$, not shown in Listing 5.11) is plotted in two different curves on the y-axis.

```
<listOfOutputs>
 <plot2D id="plot1" name="V_T (original and perturbed) over time">
  <listOfCurves>
   <curve xDataReference="d1" logX="false" yDataReference="d2"
    logY="false" />
   <curve xDataReference="d1" logX="false" yDataReference="d3"
    logY="false" />
  </listOfCurves>
 </plot2D>
</listOfOutputs>
```

Listing 5.13: Definition of a 2 dimensional plot (example)

The output will be a two-dimensional plot that shows the development of variable `v1` from model `model1` (referred to via `task1`) over time.

SED-ML examples

The complete example for the SBML version of the LeLoup model[14] has been published in [Köhn and Le Novère 2008]. A second complete example for a SED-ML compliant simulation experiment description of a CellML model, and with detailed explanation of the experiment is given in Appendix A.4.4.

Further examples for SED-ML simulation experiments created in line with the development of SED-ML support in the *SBW Workbench* are available from `http://libsedml.sourceforge.net`.

5.4.3. Conceptual design of a simulation experiment database

Extracting some of the information in SED-ML and storing it explicitly allows for indexing that data and incorporating it in the ranked search proposed in Chapter 7. A first approach to storing SED-ML information in a structured way is shown in the relation diagram in Figure 5.11. I will refer to the database as the *Simulation Experiment Database* (*sDB*).

The main relation is `simEx`. It holds an internal ID for each stored simulation experiment description (`idSimEx`) and stores both, the description file itself (`simExFile`) and its format (`simExFormat`). All information is mandatory. In addition, a short textual description of the simulation experiment might be provided (`simExDescr`).

Further information stored in *sDB* include the output type used in the experiment (i. e. 2D plots, 3D plots, table, video) and links to result data. The output type is stored in the `outputType` attribute of the `output` entity; the link to the result data is stored in the `resultDataLink` attribute of the `result` entity. That attribute ideally contains a URI to a result data file in a standard format such as SBRML (Section 3.1.1). Furthermore, the characteristic behavior of the system, as shown in the linked result data set, can be stored in the `dynamics` attribute. The use of a TeDDY ID (Section 3.1.2) is recommended.

Both, the model and the applied experiment type are pair-wise stored in the `task` entity. The `model_idModel` entity links to one of the pre-defined `modelFormat` attribute values in the `model` entity. The `experimentType_idExperimentType` links to one of the pre-defined `expTypeName` attribute values in the `experimentType` entity.

The proposed storage allows to answer some fundamental questions on the simulation experiments:

- Show all experiments for model *modelID*

[14]`urn:miriam:biomodels.db:BIOMD0000000021`, last accessed 14 March 2011.

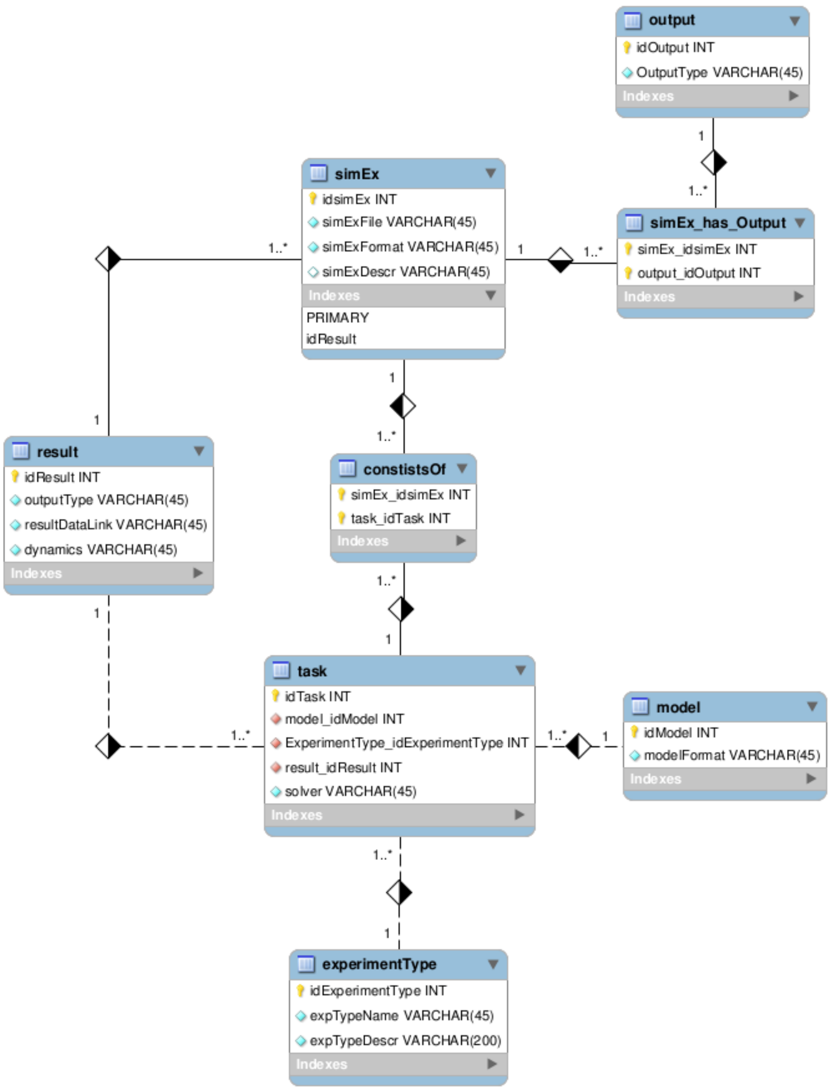


Figure 5.11.: Proposed database schema for the storage of SED-ML encoded simulation experiment descriptions.

- Show all simulation experiments for model *modelID* that have result data attached
- Show all experiments that are bifurcation analyses on model *modelID*

Combination with a model repository (*mDB*, Chapter 6) allows for even more complex queries:

- Show all models on the cell cycle that have a time course simulation experiment available, where the result data shows oscillating behavior.

The suggested structure does, however, not allow to query for individual entities used in the simulation. Queries for "simulation experiments simulating the transient concentration of `beta-catenin in the nucleus` in a time course simulation" are currently not supported.

5.5. Implementation

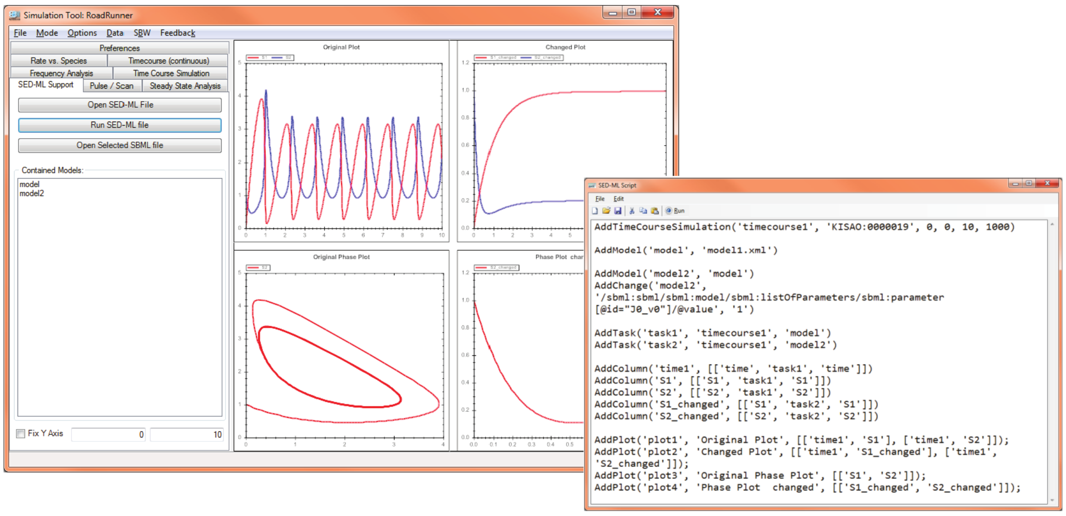
The introduced SED-ML format[15] has been implemented in an XML Schema (Appendix A.4). The current SED-ML version is *Level 1 Version 1 Release Candidate 1.* Prototype implementations of SED-ML are available in different simulation tools. In addition, SED-ML libraries for Java and C++ and a SED-ML validation tool have been developed by colleagues from different institutes.

Simulation tools The Systems Biology Workbench (SBW, [Bergmann and Sauro 2006]) is a modular framework for modelling, simulation and analysis applications in systems biology. It comes with the simulation tool *roadRunner* which supports the import and export of SED-ML files version L1V1R1. SBW internally uses a `.sedml` format that stores the model files together with the actual SED-ML description. An additional *SED-ML Script Editor* is available to change existing SED-ML files in a Python based script language. All implementation work has been done as part of the Ph.D. project of Bergmann [2010]. Figure 5.12 shows a screenshot of the SED-ML script language and the SED-ML import.

The second simulation tool implementing support for SED-ML is *Virtual Cell* (VCell, [Moraru et al. 2008]), a software tool for the analysis, modelling and simulation of cell biological processes.

JWS Online Simulator as well investigates SED-ML support. So far, only tests with semi-automatic import and export of SED-ML files have been done. Conceptually, the SED-ML format is mappable on the internal JWS Online Simulator

[15]`https://sourceforge.net/projects/sed-ml`, last accessed 15 December 2010.



```
AddTimeCourseSimulation('timecourse1', 'KISAO:0000019', 0, 0, 10, 1000)

AddModel('model', 'model1.xml')

AddModel('model2', 'model')
AddChange('model2',
'/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter
[@id="J0_v0"]/@value', '1')

AddTask('task1', 'timecourse1', 'model')
AddTask('task2', 'timecourse1', 'model2')

AddColumn('time1', [['time', 'task1', 'time']])
AddColumn('S1', [['S1', 'task1', 'S1']])
AddColumn('S2', [['S2', 'task1', 'S2']])
AddColumn('S1_changed', [['S1', 'task2', 'S1']])
AddColumn('S2_changed', [['S2', 'task2', 'S2']])

AddPlot('plot1', 'Original Plot', [['time1', 'S1'], ['time1', 'S2']]);
AddPlot('plot2', 'Changed Plot', [['time1', 'S1_changed'], ['time1',
'S2_changed']]);
AddPlot('plot3', 'Original Phase Plot', [['S1', 'S2']]);
AddPlot('plot4', 'Phase Plot  changed', [['S1_changed', 'S2_changed']]);
```

Figure 5.12.: Screenshot of the SBW simulation tool RoadRunner. Left: The SED-ML Script Editor. Right: An imported and simulated SED-ML experiment using two different models (model1 and model2). Figure taken from [Bergmann 2010, Fig. 57, bottom].

simulation description format; SED-ML support in that tool is feasible and only a matter of implementation costs.

SED-ML validator Richard Adams at the *Centre for Systems Biology at Edinburgh* developed a SED-ML validator that allows to check the validity of a given SED-ML file against the L1V1R1 schema[16].

SED-ML libraries Two libraries support application developers working with SED-ML. The first one is called *LibSedML*[17]. LibSedML implements all constructs available in SED-ML (i. e. the simulation experiments, models, model modifications, along with the description of output formats). Using this library, existing SED-ML documents can be read, and new documents can be constructed [Bergmann 2010]. The second library is called *jlibSEDML*[18]. jlibSEDML is a Java library supporting SED-ML L1V1R1. Its functionality includes the creation and reading of SED-ML files, and allows for the manipulation of encoded MathML expressions. A validator

[16]`http://www.bioinformatics.ed.ac.uk:8080/SBSVisualWebApp/html/sedml/`, last accessed 14 March 2011.

[17]`http://libsedml.sourceforge.net/`. LibSedML has been implemented in C# by F. Bergmann. Last accessed 14 March 2011.

[18]`http://jlibsedml.sourceforge.net/`. jlibSEDML has been developed by I. Moraru, R. Adams, D. Vasilescu and A. Lakshminarayana and is available under the MIT license. Last accessed 14 March 2011.

is integrated as well.

The CellML group works on a *SED-ML Processing Service* (SProS).

SED-ML in BioModels Database Li et al. [2010] mention the distribution of SED-ML files together with the models as one of their future aims.

5.6. Summary

Results

This chapter introduced a Minimum Information guideline (MIASE) which in a general, format-independent way defines the minimum information for simulation experiments in the area of computational biology. Following the investigations on MIASE, the SED-ML format for the encoding of such simulation experiments has been developed. Figure 3.1 on page 44 shows how the developed formats integrate with existing standardisation efforts.

The main achievement is the possibility to store, export, import and exchange simple simulation experiments in a standardized, XML-based way. The information encoded in SED-ML can be incorporated in the ranked retrieval process and as such extends the search possibilities by what is referred to as *experiment-related meta-information.* Besides that, SED-ML in general is a helpful tool for the reuse of simulation experiments performed on a bio-model. Its existence saves time and effort in running a model, may it be for educational purposes, model curation or experiment repetition.

The use of SED-ML for the latter tasks has been shown in different library implementations which are included in running simulation tools. The extraction of meta-information available from SED-ML, and how that information can be stored in a relational database has furthermore been outlined.

Discussion

MIASE compliance in SED-ML SED-ML respects the MIASE guidelines. Table 5.1 shows a mapping of MIASE rules (numbers) on SED-ML concepts (XML element names). The generic simulation experiment examples used in Section 5.2.3 are linked to the different concepts for clarification.

The current version of SED-ML does not fully encode all information requested by MIASE. Especially the information about dependencies on the used simulation environment and platform is currently not encoded in SED-ML (**rule 2d**). While links to the specific algorithms have to be provided through a KiSAO ID, not all

rule	SED-ML element/attribute	sample code
1a	`model` (ll. 167-181)	Listing 5.2
1b	`model` (ll. 167-181), `listOfChanges` (ll. 298-312)	Listings 5.2, 5.4
1c	`language` (l. 174)	Listing 5.2
1d	`listOfChanges` (ll. 298-312)	Listing 5.4
2a*	`task` (ll. 114-127), `algorithm` (ll. 85-94)	Listings 5.6, 5.8
2b	`uniformTimeCourse` (ll. 95-113)	Listing 5.6
2c*	`algorithm` (ll. 86-94)	Listing 5.6
2d*	-	-
3a	`dataGenerator` (ll. 380-394)	Listing 5.10
3b	`listOfOutputs` (ll. 229-241)	Listing 5.12

Table 5.1.: Mapping of MIASE rules sorted by number (first column) on SED-ML XML element names and attribute names (second column). The line numbers comply with the schema definition in Appendix A.4. The third column refers back to the sample listings showing the corresponding generic code for a SED-ML instance. Some MIASE rules are not fully supported by SED-ML; they are marked with an asterix (*).

algorithms are available: information on how to implement those algorithms are not demanded by SED-ML (**rule 2c**). Furthermore, the definition of nested simulations with data processing in between the simulation setups as well as the definition of a particular order of simulation steps are not supported in the current SED-ML version (**rule 2a**). They are, however, planned for the future.

Reuse and exchange of simulation experiment descriptions The major idea of SED-ML is the standardized encoding of simulation experiments, thereby improving bio-model reuse. In this process, not only the models are of relevance; equally important is the ability to reuse simulation experiments applied to them.

SED-ML enables the *exchange* of simulation experiment descriptions between different work groups, using similar or different platforms. As a proof of concept, SED-ML has been used to exchange simulation setups between the simulation tools *Roadrunner* and *JWS online simulator*. A simulation experiment has been performed in the JWS online simulator, the SED-ML file has been exported and then been re-loaded into the SBW simulator RoadRunner. The same simulation result was obtained for a time course simulation[19].

[19]2010 CellML Annual workshop, Auckland, `http://www.cellml.org/community/events/workshop/2010/presentations/Waltemath_sedml.pdf`, last accessed 07 February 2010.

Use of simulation experiment information for model retrieval Information encoded in SED-ML can be incorporated in the ranked retrieval process. Section 5.4.3 showed a database schema proposal extracting relevant information about a simulation experiment. That information can be used for the ranking introduced in Chapter 7.

Although MIASE and SED-ML already proved to be applicable, there are a number of limitations and open tasks.

Generality of MIASE and SED-ML MIASE is currently restricted to certain application fields within systems biology. In the long run, we wish to define MIASE in a way that applies to a broader range of simulation experiments and models. It can be expected that a similar development to the one taken by MIAME will take place where "its widespread adoption by scientific journals also exposed some of its weaknesses, including the need to develop domain-specific extensions" [Quackenbush 2006].

The current SED-ML version only covers *time course experiments*. One goal is to extend SED-ML towards other experiment types such as bifurcation analyses. SED-ML has so far been applied to models encoded in SBML[20] and CellML. We wish to extend the evaluation of SED-ML on other representation formats. Ongoing discussions address the application of SED-ML on the encoding of simulation experiments in the neuroscience community, using both NeuroML and NineML[21]. For example, researchers from the *Computational Neurobiology* group at the EBI (Hinxton) and colleagues from the *Adaptive and Regenerative Software Systems* group at Rostock University aim at testing SED-ML for the description of neuronal network simulation.

Extending KiSAO KiSAO needs to be extended by concepts that better describe the covered simulation algorithms. For example, each algorithm has a number of defined characteristics, or parameters, that need to be applied before simulation (e. g. the step size in a Runge-Kutta-based simulation, or upper and lower limits). The parameters must be studied for the different algorithms, and then be formalized in a way that they can be used with the different algorithms. The *encoding* of such characteristics is not trivial, neither is the identification and generalization of the particular parameters – a task that will require domain knowledge. Moving to OWL

[20]`http://libsedml.svn.sourceforge.net/viewvc/libsedml/trunk/Samples/` provides a list of examples, last accessed 29 September 2010.
[21]`http://www.nineml.org/`, last accessed 17 March 2011.

instead of OBO format is considered for the next version to allow for more complex relationship types between classes.

Model evolution SED-ML files use the `source` attribute to refer to external resources containing the model code for simulation. While the repositories do provide stable and standardized URN's for their models, they do not yet distinguish between *different versions of a model referenced from a SED-ML file.*

For example, an early SED-ML description pointed to the Repressilator model available from BioModels Database (`source="urn:miriam:biomodels.db:BIOMD0000000012"`). However, since then the Repressilator model representation has been updated by adding an additional parameter and an assignment rule to the model, and normalizing each reaction. This change already influences the outcome of the SED-ML simulation, if undiscovered. A thorough versioning concept that will allow SED-ML users to refer to a model in a particular version is needed to overcome this problem. Bio-model versioning is further discussed in Section 6.4. Unfortunately, the proposed solution does not help the SED-ML problem as long as a unique identifier for each model version is missing in the reference system.

SED-ML validity SED-ML, as it stands at the moment, allows for applying changes to the model before simulation, as long as the change addresses an XML element or attribute in the model with a valid XPath expression. Additionally,

- the `ComputeChange` contains a valid piece of `mathML`, as well as it defines the used variables by valid XPath references and
- the `newXML` element contains a valid piece of XML.

For the moment, none of the conditions are checked when a SED-ML file is loaded into a simulation environment as SED-ML files are intended to be automatically created by the simulation tool. Simple checks for the validity of the exported file against the SED-ML XML schema are possible using the aforementioned SED-ML validator.

However, for further checks on a simulation experiment description, it will be useful to have means for validating changes on a model before simulation. One application is the versioning of available models in online resources; a valid SED-ML file might become invalid if the models referred to by the SED-ML description evolve in a way that is incompliant with the SED-ML modifications. For example, the extract of a SED-ML model change specification given in Example 5.5 updates the attribute value of `V_mT` to the new value `0.28`. The XPath expression `/sbml/model/listOfParameters/parameter[@id='V_mT']/@value` identifies the attribute inside the referenced model (i. e. it assumes the existence of a piece of XML code as shown in Listing 5.14).

```
<sbml>
 <model>
  <listOfParameters>
   <parameter id="V\_mT" value="SOME VALUE" />
   [..]
  </listOfParameters>
  [..]
 </model>
</sbml>
```

Listing 5.14: Extract of the updated model code leading to an invalid SED-ML description

A change in the referenced model (e. g. the update to the model specification given in Listing 5.15) will lead to incorrectness of the SED-ML specification file.

```
<sbml>
 <model>
  <listOfParameters>
   <parameter id="V" value="SOME VALUE" />
   [..]
  </listOfParameters>
  [..]
 </model>
</sbml>
```

Listing 5.15: Extract of the model code that is perturbed by the SED-ML file extract given in Listing 5.5

The updated model extract has a changed parameter `id` (either it is changed, or the parameter with `id=V_mT` is deleted and a new parameter with `id=V` is introduced). As a consequence, the XPath expression pointing to the parameter with `id=V_mT` is invalid. The value cannot be updated and the simulation experiment cannot be reproduced.

The same problem occurs at any point of the SED-ML file specification where XPath is used, for example it can also lead to inconsistency in the output of a simulation. A first step towards avoiding invalid SED-ML files due to model evolution is a check on all given XPath expressions and specified MathML snippets before simulation. This service could be provided externally by the SED-ML community. One possibility to realize the validation is a simple XPath expression validator.

Explicit variables in SED-ML One limitation of SED-ML is that the XPath concept only allows to address explicitly defined model entities (using an XPath expression). While the concept of implicit variables solves problems such as defining time, there is no solution for the explicit addressing of implicitly modeled entities. For example, some models allow to generate pools of entities (e. g. a pool of 1000 neuronal cells) at run time; these entities are not explicitly represented in the bio-model's XML representation. Therefore, it is not possible with the current implementation of SED-ML to address one instance out of the entity pool.

6. Format-independent model storage

After all, a rose by any other name is still a rose; you just cannot find it in the database.

(Quackenbush [2006])

Adequate bio-model storage is a prerequisite for later applications, including model retrieval, model management, model coupling, or model visualisation.

The following chapter investigates storage and versioning solutions for bio-models and introduces an annotation-based bio-model storage approach. After a problem statement (Section 6.1), prerequisites for the annotation-based bio-model storage approach are described in Section 6.2. The proposed storage, including the developed database schema, is introduced in Section 6.3. It is shown how the combination of black-box storage and additionally extracted semantics provides an appropriate information base for later model retrieval. Section 6.4 then discusses model versioning as one improvement of the developed system over existing solutions. The summary of implementations (Section 6.5) is followed by main results and future prospects (Section 6.6).

The result is a database schema for bio-model storage [Köhn et al. 2009; Waltemath et al. 2011c]. The schema is generic and therefore can be used with different representation formats. The database supports bio-model versioning [Hälke et al. 2011] and is currently being evaluated for the use of a next-generation bio-models storage and maintenance system developed in cooperation at the *Deutsches Krebsforschungszentrum* (Germany) and the European Bioinformatics Institute (UK)[1].

6.1. Problem statement

While model representation formats enable model exchange among tools and researchers, they are not sufficient to determine the models' nature [Henkel et al. 2011]. Model representations are mostly automatically created from software, resulting in meaningless XML element and attribute names such as `s1` to `sn` for encoded species,

[1]Just a Model Management Platform, `https://bitbucket.org/jummp/`, last accessed 18 February 2011.

or `r1` to `rn` for encoded reactions [Henkel et al. 2010]. Models that are created from literature use convenient and biologically non-meaningful names that cannot be associated directly with a particular biological function or entity [Li et al. 2010]; the strings do not allow for inference of any information on the constituent's meaning. Even if the modeler provides real names for species and reactions through the XML element and attribute names, the information content is limited. Strings are affine to typing mistakes. Different languages result in different names for the same semantic concept. At the same time, many cases for synonyms and hynonyms can be found.

When aiming at model retrieval, XML instance matching techniques (such as the ones proposed in [Miller et al. 2001; Engmann and Massmann 2007]) are not sufficient. Meta-information have to be incorporated in the retrieval process [Henkel et al. 2010]. Prerequisites are the *encoding* of information in the bio-model and a suitable *storage concept* to maintain that information. The standardized encoding and annotation of bio-models with meta-information (Sections 2.3 and 4.2) addressed the first problem. Existing model repositories (Section 3.3) aim at solving the second one. With annotated bio-models being encoded in different model representation formats several model repositories evolved. Opposed to the model representation in XML, the encoded meta-information is *independent* of the underlying structure. Although there exist different approaches for meta-information encoding (Section 3.2) they all are based on RDF. This suggests to construct a model storage system using a formalism-independent, meta-information based apporach. The feature of separating model and meta-information then allows for efficient retrieval and comparison of models independent of the model encoding [Köhn et al. 2009].

Such an approach also facilitates better model management within international projects. International modeling projects cooperating on large-scale development of biological systems in Europe and world wide are of high research interest (e. g. the SysMO project that is concerned with systems biology for micro organisms `http://www.sysmo.net/` or the BaSysBio project that is concerned with Bacillus Systems Biology `http://www.basysbio.eu/`). The projects involve groups from different countries. For example, 15 European Research groups and one Australian university are involved in the BaSysBio project. Major database-related tasks include the maintenance of models, their versions and variants. Often, the developed models are stored and maintained in distributed repositories such as Wikis, Content Management Systems, SVNs or just local stores, making them available to colleagues on demand. This situation easily leads to update problems, and it hampers the models' reuse. Solutions for bio-model management are needed, in particular for bio-model versioning. Model versions are being created during the development process and, in addition, modified during the curation process to justify the reference publication or to erase modeling and annotation errors. A storage system should adhere to both, changes in the model code and in the annotations. Current model repositories, however, lack thorough versioning concepts. Although models in existing repositories change both in structure and annotation (and therefore potentially in meaning), users are not notified about these updates. They can neither follow nor recapitulate a model's development over time. Explanations of why changes had been applied are missing. As a consequence, a user cannot rely on a model resource to be stable; he cannot address a particular version of a model. Changes in the model might also affect other models built on top of a particular model in a particular version. To change that situation, a detailed version system must be developed and then be coupled with the model storage system.

6.2. Prerequisites

The following discussions assume the existence of annotated bio-models in standardized, XML-encoded representation formats. Although a solution for models of different types of biological systems is desirable, the main focus of this work is on the storage of biochemical cell models as those represent the majority of models developed today. This is justified in the comparably large number of bio-models for chemical reaction networks available online for testing.

As shown in Section 3.2, sophisticated solutions for the encoding of model meta-information exist. While mathematical models are often annotated, and some journals even require annotation information to be present, the situation is different for discrete-event based modeling. In that community, a widely-accepted way of providing meta-information for bio-models in a computer readable standardized manner is missing.

Section 6.2.1 extends the previously given definition of bio-model; a detailed example for the encoding of a model is given. Identified meta-information for bio-models has been categorised in a coordinate system (Section 6.2.2). Section 6.2.3 introduces the idea of a generic parser for the information extraction from bio-model files. The export format of the parser which is also used for the storage and ranked retrieval is described in Section 6.2.4.

6.2.1. Annotated bio-models

The following chapters necessitate an extension of Definition 2.1.4 to distinguish bio-model source code, bio-model annotations, and XML comments.

Definition 6.2.1 (Annotated Bio-model (adapted from [Hälke 2010])). *An* annotated bio-model *(in the following also bio-model)* m_{bio} *is a computational model of a biological system that allows for an explanation of the mechanism behind the observed behavior of the biological system. The bio-model is considered to be annotated with meta-information.* m_{bio} *then can be described as a tuple* $m_{bio} = (m_S, m_A, m_C)$ *of*

1. *model source code* m_S *in a machine-readable format,*
2. *in XML encoded annotation information* m_A *describing the biological nature of the bio-model and of it's constituents, and*
3. *in XML encoded comments* m_C*.*

A distinction between annotation information of the model itself and of its constituents will not be made, unless stated otherwise. All following concepts assume the existence of *annotated bio-models in a representation format.* Appendix A.3.1 summarises the steps from modeling a biochemical reaction to encoding it in an annotated bio-model, using the example of modeling an enzymatic reaction.

6.2.2. The model information coordinate system

Figure 3.1 on page 44 outlined the kinds of meta-information with respect to the different *levels* on which meta-information can be defined. To classify the diverse bio-model data and information, Figure 6.1 organizes the different *types* in a coordinate system. An adopted and simplified version of the Leloup and Goldbeter [1999] model[2] has been used to clarify the approach. The SBML code is available from Appendix A.2.1.

On the one hand, a distinction between model and meta-model is proposed (y-axis). The *model* is the actual XML code m_S. The *meta-model* is formed based on the annotation information m_A given for a model. On the other hand, a distinction between structure and data is shown (x-axis). For the scope of this work, the *structure* is given by the defined reactions of a bio-model (i. e. its network representation) or its meta-model respectively. The *data* includes all information available from the description of the model, or the meta-model respectively.

A *model on structure level* is the XML document tree resulting from the XML representation of a bio-model. Formats supporting that representation have been mentioned earlier in this work and include SBML, CellML or πML (Section 2.2). For example, the structure in Figure 6.1 shows two unary reactions: species `T0`

[2] `urn:miriam:biomodels.db:BIOMD0000000021`, last accessed 27 September 2010.

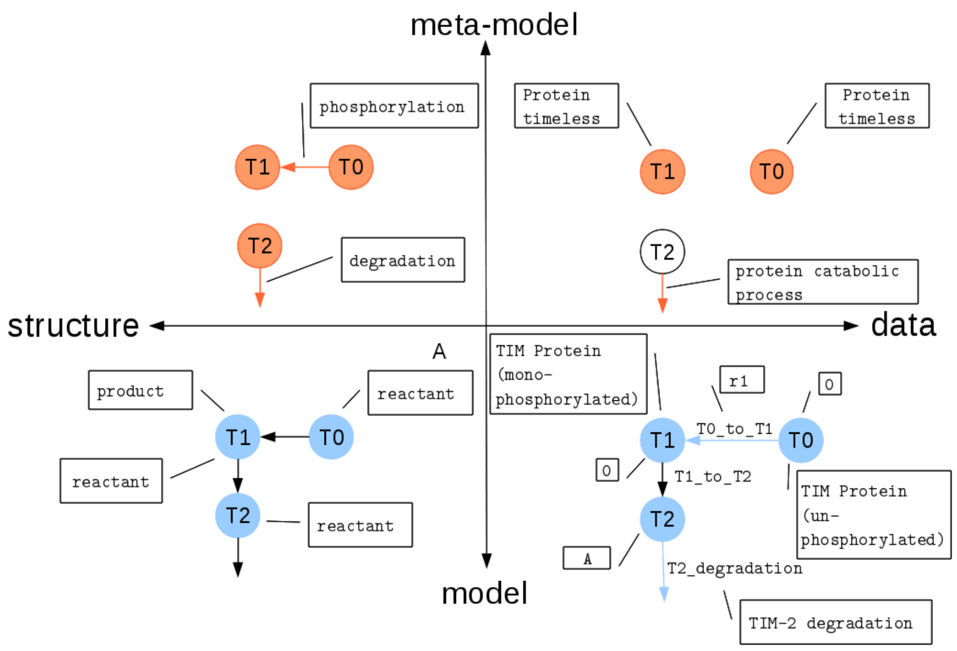


Figure 6.1.: Dimensions for information sources: Data, model, structure, meta-model. The model encoding structure (inferred from the XML documents) is shown in blue, the model annotation structure (inferred from the annotations) is shown in orange.

as a `reactant` is transformed into species `T1` which is the reaction `product`, and species `T1` playing the role of a `reactant` is then transformed into the `product T2`.

A *model on data level* represents the data encoded in the model, including constituent names, and a particular parametrization (i. e. initial values of the entities). Examples from SBML are the species names, model parameters, and compartment sizes. Some formats might prefer to provide concrete parametrization of the model constituents in a separate file. They are nevertheless regarded data for this work. The model on data level as shown in Figure 6.1 contains three species "TIM protein (unphosphorylated)", "TIM protein (mono-phosphorylated)", and "A". Furthermore the initial concentrations for `T0` and `T1` are given (0). Finally, the reaction between `T0` and `T1` is named "r1", and the reaction `T2` is named "TIM-2 degradation". These constituent names are provided in the model code. They are not necessarily meaningful; often they are created automatically by modeling tools, leading to namings such as "r1", or "A".

A *meta-model on structure level* is built based on the annotations provided with a model and concerned with the function of its entities. The resulting connectivity graph does not have to correspond with the document tree, as models are not necessarily fully annotated. In the given example, the reaction between `T1` and codeT2 is not annotated, and as such not included in the meta-model. Annotations on structure level describe the functionality of the single structural elements of the according bio-model. For example, in Figure 6.1 `r1` is identified as a `phosphorylation`, and `TIM-2 degradation` (the name of the reaction constituent) is identified as a `degradation`. Typically, those information are provided by the SBO term (Section 3.1.2) associated to a model constituent.

A *meta-model on data level* is built based on the annotations provided with a model and concerned with the biological meaning of its entities. The information can technically be encoded in the same way as the meta-model information on structure level, but it conceptually differs in the perspective it describes the model. In Figure 6.1, the species `T0` and `T1` are both annotated as the biological entities `Protein timeless`. Another example to show the difference of meta-model annotation on structure and data level is the predator-prey model. All information on the model side may stay the same, and also the meta-model structure may be identical. But a change in the meta-model on data level can change that bio-model's biological reference: Imagine a change in annotation of the species from "rabbit" to "zebrafish" and from "fox" to "shark".

In addition to the meta-information that is encoded in common representation formats, other model characteristics such as model behavior or model scope (Section 2.1) need to be considered. Ongoing research investigates their encoding in bio-ontologies (Section 3.1.2).

6.2.3. Model parser

The necessary information for *mDB*, in particular the model meta-information, needs to be extracted from the bio-model encoding. As this work aims at a format-independent storage approach, a *general model parser* is desirable. However, existing representation formats use different XML schemas for encoding their bio-models. Even though many do make use of annotations in rdf:RDF format, the way that they implement annotation support varies (Section 3.2). This section introduces a model parser that is capable of extracting the relevant information from a bio-model. It is based on a modular set of XSLT stylesheets. While the current focus is on SBML model import, the parser can be extended towards the support of other formats in the future.

Existing solutions to parse SBML files include: (1) the *libSBML*, an extensive C++ library to work on SBML files that is developed by the SBML community, (2) JSBML[3], is a library similar to libSBML, but written in Java, and (3) the *libAn-*

[3]`http://jsbml.sourceforge.net/`, last accessed 14 January 2011.

notationSBML [Swainston and Mendes 2009], a library particularly working with annotations of SBML files. After evaluation of libSBML and libAnnotationSBML[4], the decision was made for developing a new parser specifically for the import of annotations from existing models in common representation formats [Hälke 2009]. The so-called *tiny-parser* provides two different output types: (1) an XML format, also referred to as the *internal representation format* (IRF, Section 6.2.4), and (2) a Java object representation for use in software tools.

6.2.4. Internal representation format

The internal representation format (IRF) is provided as an XML Schema definition (Appendix A.5.1). IRF instances are used to import the model meta-information and data into *mDB*. A detailed description of the IRF-to-*mDB* mapping follows in Section A.5.2. The IRF furthermore represents the internal exchange format within the *Sombi* framework (Section 7.4); it forms the information base for the ranked retrieval functions and for the front-end implementation. Section A.5.2 also provides the mapping of IRF concepts on the features used for the ranked retrieval.

The structure of the IRF relates to the SBML structure, due to the fact that the main application area of this work is the retrieval of biochemical models, with most examples being encoded in SBML.

Introducing an intermediate layer between representation formats and applications (*mDB*, ranking, front-end) has certain advantages. The IRF is a stable interface that allows developers of different representation formats (and different versions of the formats) to write their own mappers. It is presumably easier to map on another XML format than to map on a database schema that is, in addition, likely to evolve over time. The ranked retrieval benefits from the IRF representation as the implementation is only accessing the IRF files and therefore does not have to be updated for each new supported model representation format. The user front-end as well stays independent of the underlying model representation format.

IRF and SBML. An extract of an IRF file representing parts of the information parsed from an SBML model is given in Listing 6.1. The listing will be used to explain the major building blocks of the IRF.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<model>
 <nameID>Gardner1998_CellCycle_Goldbeter</nameID>
 <name />
 <dbModelID />
 <dateCreated>2005-01-30T13:02:57Z</dateCreated>
```

[4]JSBML was not available at that time.

```
<dateModified>2009-08-10T15:44:34Z</dateModified>
<dateSubmitted>2010-10-03T09:16:51.609Z</dateSubmitted>
<stable>true</stable>
<modelDescription>
     This a model from the article: A theory for controlling cell cycle
         dynamics using a reversibly binding inhibitor. [..]
     Abstract: We demonstrate, by using [..]
</modelDescription>
<modelFormalism>
 <type>urn:sedml:language:sbml.level-2.version-4</type>
</modelFormalism>
<modelFile>
 <fileLocation [..] />
 <fileVersion [..] />
 <modelFileContent [..] >
 <annotationFileContent [..] />
 <experiment/>
</modelFile>
<listOfQualifiers>
 <modelQualifier modelQualifierRole="is">
  <URI>urn:miriam:biomodels.db:BIOMD0000000008</URI>
  <URI>urn:miriam:biomodels.db:MODEL6614879888</URI>
 </modelQualifier>
 [..]
</listOfQualifiers>
<listOfKeywords/>
<listOfAuthors>
 <author authorRole="creator" main="false">
  <firstName>Bruce</firstName>
  <lastName>Shapiro</lastName>
  <organisation>NASA Jet Propulsion Laboratory</organisation>
  <email />
 </author>
 <author authorRole="publisher" main="true">
  <firstName>T S</firstName>
  <lastName>Gardner</lastName>
  <organisation>Center for BioDynamics and Department [..]
  </organisation>
  <email />
 </author>
 <author authorRole="submitter" main="false" [..]/>
 [..]
</listOfAuthors>
<referenceDescription>
 <referenceURI>urn:miriam:pubmed:9826676</referenceURI>
 <referenceTitle>A theory for controlling cell cycle dynamics using a
     reversibly binding inhibitor.</referenceTitle>
 <referenceAbstract>We demonstrate, by [..] </referenceAbstract>
</referenceDescription>
<distributionStatement />
<listOfCompartments>
 <compartment>
  <id>Cell</id>
  <metaid>_202836</metaid>
```

```
59     <name>Cell</name>
60     <notes />
61     <bioQualifier bioQualifierRole="identity">
62      <URI>urn:miriam:obo.go:GO%3A0005737</URI>
63     </bioQualifier>
64    </compartment>
65   </listOfCompartments>
66   <listOfSpecies [..] />
67   <listOfReactions [..] />
68 </model>
```

Listing 6.1: The internal representation format, with an extract of the information parsed from an SBML model (`urn:miriam:biomodels.db:BIOMD0000000008`)

Most important is the information on the model itself, including the model name, ID, internal databaseID, stability information about the model, and creation and modification dates (ll. 3-9). Another important aspect of information is that about the people involved in the model development and storage process, referred to as authors. They may take different roles, such as creator or publisher (ll. 32-46). The IRF also encodes information on the reference description, including the link to that publication (l. 49) and more detailed information on the publication title and the abstract (ll. 50-51). Finally, the major part of the document consists of model meta-information extracted from the annotations of model elements, including compartments, species and reactions (ll. 54-66). The ontology references are stored together with the corresponding qualifiers.

Information that cannot automatically be extracted from the model representation files needs to be added either by the user (e. g. through a web interface) or determined by the system. It includes the model submitters (l. 45) and version information (l. 19).

IRF and other representation formats. The IRF as it stands now is capable of storing the relevant SBML meta-information. With slight extensions in the tiny-parser's XSLT scripts it can also be used to extract annotations from CellML or πML models. The simple πML example given in Listing A.7 on page 201, for example, can be represented in the internal representation format as exemplified in Listing 6.2.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<model>
  <nameID>nameID46</nameID>
  <name>euglena.xml</name>
 <dbModelID [..] />
 <dateCreated>2005-01-30T13:02:57Z</dateCreated>
 <dateModified>2009-08-10T15:44:34Z</dateModified>
 <dateSubmitted>2010-10-03T09:16:51.609Z</dateSubmitted>
 <stable>false</stable>
```

```
<modelDescription [..] />
<modelFormalism>
   <type>PiML</type>
</modelFormalism>
<modelFile [..] />
<listOfQualifiers>
 <bioQualifier bioQualifierRole="identity">
  <URI>urn:miriam:taxonomy:3038</URI>
 </bioQualifier>
 [..]
</listOfQualifiers>
<listOfKeywords [..]/>
<listOfAuthors>
 <author authorRole="creator" main="false">
  <firstName>Robert</firstName>
  <lastName>Kuehn</lastName>
  <organisation>Rostock University</organisation>
  <email [..] />
 </author>
 [..]
</listOfAuthors>
<referenceDescription [..] />
<distributionStatement [..] />
<listOfCompartments [..] />
<listOfSpecies>
 <species>
  <id [..] />
  <metaid>metaid_000005</metaid>
  <name>euglenaSpecies</name>
  <notes />
  <bioQualifier bioQualifierRole="identity">
   <URI>urn:miriam:taxonomy:3038</URI>
  </bioQualifier>
 </species>
</listOfSpecies>
<listOfReactions>
 <reaction>
  <id [..] />
  <metaid>metaid_000004</metaid>
  <name>phot</name>
  <bioQualifier bioQualifierRole="identity">
   <URI>urn:miriam:obo.go:GO%3A0015979</URI>
  </bioQualifier>
  <bioQualifier bioQualifierRole="identity">
   <URI>urn:miriam:obo.go:GO%3A0019684</URI>
  </bioQualifier>
 </reaction>
</listOfReactions>
</model>
```

Listing 6.2: The internal representation format, encoding information from the sample πML model shown in Listing A.7.

6.3. Model storage

The identified meta-information must be stored in a way that allows for easy access and indexing, thereby improving tasks such as model retrieval, visualization, coupling, or merging.

Model repositories differ in how they store bio-models and to what extend they respect standardization efforts. Some drawbacks have been mentioned in the introduction to this chapter. How to solve the problem of differing meta-information standards has been discussed in Chapter 4.

The storage approach introduced in this section aims at solving the two following problems:

- Existing databases are autonomous, partially heterogeneous and not integrated. Most repositories are specialized on storing models of one specific model representation format. (An exception is the ModelDB which follows a format-independent storage approach that is based on a small set of user-given meta-information.)
- Existing model repositories lack sophisticated model version control mechanisms. This situation is problematic when referencing models in a reuse scenario.

6.3.1. Considered storage approaches

The fast development of XML-based standards for biology models over the last years is reflected in more than 80 XML-based standards of various interest having been available for the community in 2006 [Strömbäck et al. 2007]. Of these, 14 were of common use for the export of data from databases or the import of data to various analysis or simulation tools. Most formats are XML-based (e. g. the introduced SBML or CellML formats), but many are also OWL-based and integrate ontology information directly, for example BioPAX. To store different XML formats commonly (e. g. files validating against the CellML Schema, and files validating against the various versions of SBML Schemas) a *mapping* on a more general schema is necessary – attributing to all concepts implemented in the different languages. A probable solution should even consider the storage of different XML-related formats, such as including BioPAX models specified in OWL. Furthermore, a common query technique for all formalisms should be available. During the investigations of this work the following options have been considered:

- Converting different formats into a “super-formalism” [Köhn and Strömbäck 2008]

- Object-oriented database storage approach [Ou 2009]
- Federated database approach
- Hybrid XML storage
- "Black Box" storage + meta-information [Köhn 2009]

The different approaches will be discussed in the following.

Conversion of different formats into one "Super-Formalism" One option to overcome the structural gap between the different standards is the provision of a mapping of different standards either on each other, or on a common format.

The attempt to convert formats into each other (e. g. SBML $\leftrightarrow$ PNML, SBML $\leftrightarrow$ CellML) leads to specific transformation scripts that are valid for particular languages versions but do not provide a generic solution [Köhn 2009].

A common "super-formalism" in OWL format is a second solution. Similarities between XML Schema and OWL model include the fact that both languages were designed to define common vocabularies and structures to support information exchange. They both build upon data type definitions (i. e. `xsd:simpleType` and `xsd:complexType` in the XML Schema versus `owl:DatatypeProperty` and `owl:ObjectProperty` in the OWL model). However, OWL offers a wide range of relations between classes to chose from (i. e. `subClassOf`, `intersectionOf`, and `unionOf`). Such a clear definition of relationships between classes (or, as they are called in XML Schema, types) is not possible in XML Schema. There, relations are restricted to IS_A- and PART_OF-relations; statements such as "Element E1 is the contrary of element E2." are not supported. The only way of defining semantics in XML Schema is the hierarchical composition of elements and type definitions, resulting in an "implicit" definition of the semantics.

In [Köhn and Strömbäck 2008], we used the little implicit semantics to define a mapping of XML concepts into OWL concepts for the specific domain of pathway standards. The initial idea for the storage of bio-models was to shred the created OWL representation-based models into a relational database. However, this approach showed several disadvantages. As the different representation formats cover different areas of biology on different granularities, a mapping of those standards is either incomplete, or results in a large schema with a low percentage of matching concepts. Even if a mapping can be defined, it heavily relies on the maintenance of the transformation scripts as standards evolve[5]. Furthermore, the transforma-

[5]For example, SBML has been available in 9 different levels and versions since the issue of SBML Level 1 Version 1 in March 2001.

tion loses information, for example on the model structure, which cannot be taken into account during the search process. Last but not least, the approach does not consider the model meta-information. Therefore, the information available for other tasks is very limited. A matching on OWL-level has been evaluated in [Köhn and Strömbäck 2008] and did not show the desired results.

Object-oriented database storage approach The SBML schema consists mainly of references between SBML elements; the XML structure is therefore relatively flat. The majority of SBML elements does not contain any textual content, but rather various attributes with corresponding values. Many attributes and elements are optional. These characteristics suggest an object-oriented approach to SBML model storage instead of using a relational one [Jung et al. 2006; Ou 2009]. Storing SBML models in an object-oriented structure enables *simple queries* such as "Find reaction products named 'C'.", *associative queries* (i. e. queries incorporating associative relations) such as "Find reactions named 'Aldolase' in the compartment 'glycosome'.", and *complex queries* such as "Find all models, their according species and compartments, in which there exist structures for reactions that convert 'ADP' into 'ATP'." [Ou 2009]. However, the meta-information is not directly encoded in the model, but has to be resolved from external resources. Therefore this approach as well does not allow to incorporate meta-information in the search. The reconstruction of the original models is not possible from the object structure. Also, when considering a format-independent storage approach, the object-structure of the different representation formats has to be mapped on a general one, which will lose capabilities in the different types of queries mentioned above.

Federated databases One approach to maintain existing bio-model files of different formats and resources is a *federated database system* (FDBS, [Sheth and Larson 1990]), defined as a collection of *cooperating* database systems that are autonomous and possibly heterogeneous. It consists of so-called component databases that are, while autonomous, participating in a federation to allow partial and controlled sharing of their data. The degree of integration may vary, depending on the chosen architecture [Sheth and Larson 1990]. However, one of the crucial aspects of an FDBS is cooperation, that is the fact "that a component DBS can continue its local operations and at the same time participate in a federation".

The current situation shows very different approaches to model storage. Most model repositories do not make use of the standard database system features. They are rather storage containers than fully fledged database systems. With one leading model repository in the area, BioModels Database, it seemed hard to build a federation of equal databases under the administration of a third party. On the other hand, the integration of the component databases did not seem to be possible on the side of the model repository developers, due to very limited work force. Another reason to decide against the use of a FDBS lies within the fact that most capabilities for fine-grained annotation-based model storage and versioning were not existent for the majority of formats considered for storage. A development of a new system for testing purposes did therefore seem to be the most promising solution.

Hybrid XML storage Another option is the relational storage of models (as XML type), with additional extraction of the identified model formalism components which are then stored in their XML-format using the provided XML datatype provided by the DBMS.

However, the major advantage of current representation formats is their rich semantic annotation. These meta-information are not part of the XML code and therefore need to be extracted and stored separately. Information gained from the XML-representation of model parts is very limited and does not justify the storage effort.

Black-box storage + format-independent meta information This storage approach is based on an abstraction of the actual model representation format, relying on the available meta-information. Doing so, the generalized storage approach suits different model representation formats such as SBML, CellML or πML [Köhn et al. 2009]. Example A.7 showed that the proposed annotation format introduced in Chapter 4 does not depend on the underlying model representation format. As such, a storage approach based on the model meta-information is applicable to XML-based model representation formats that follow the meta-information provision scheme in general[6].

6.3.2. Model database mDB

I propose to extract data, meta-data and meta-information from bio-model representations (Definition 6.2.1) and store them in a relational database called *mDB*. Models are imported by parsing the model representation formats and creating the IRF (Section 6.2.4) which can be mapped on the *mDB* schema as shown in Figure 6.2. The mappings are defined in Section 7.4.

The transformation, again, is not loss-less – only data, meta-data and meta-information regarded necessary for the later retrieval and maintenance of bio-models

[6]This is given the different communities can agree on a common annotation scheme.

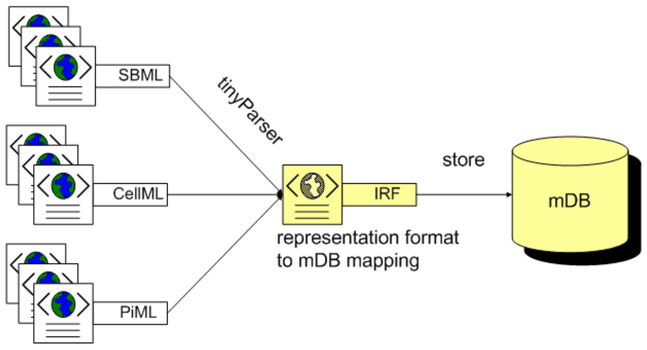


Figure 6.2.: The import process of model representations in XML format into the *mDB* via an internal representation format.

are extracted from the XML representations. However, the approach allows for the comparison of models of different formats on a common base. As the original model representation files are as well kept (black box storage), the system can be extended towards further queries, for example by using XQuery to query the XML files directly. The extracted meta-data includes some information on the model's network structure and allows for later structural queries.

Bio-model XML- representation The physical model file is stored in a black-box manner. However, the retrieval of models based on the syntactical structure of the model encoding formats does not provide sufficient information. Even models with a similar XML structure may differ in their meaning. The model's semantics plays an important role in the determination of model similarities. By using the annotations for model search, the system design is not dependent on a certain model format.

Nevertheless, the model file needs to be kept in the system for later retrieval, as well as for the determination of diffs in the versioning process, and for structure queries not supported by the *mDB* structure itself. To realize structure queries, languages such as *XQuery* will be used in the future. XQuery [Boag et al. 2010] is an ad-hoc query language for XML documents based on XPath.

Model and structure data The model data (Figure 6.1 on page 125) extracted from the bio-model representation includes the names of the model constituents as encoded in the XML file. These constituents are stored as model *components* in *mDB* and have a specific *role* attached (e. g. species, reaction, or event).

The structure (i. e. the relation between different model components) is in addition encoded through further relations in *mDB*. For example, the `participatesIn` relation encodes the relation between a species and a reaction component. The species

Meta-information	Meta-information type	Value for ranking
Model name	content-independent	low
*Creator	content-independent	low
*name, email	content-independent	low
main contact, organization	content-independent	low
*Creation date	content-independent	low
*Reference description	domain-specific	high
Distribution statement	content-independent	low
Keyword	content-descriptive	medium
*Formalism	domain-specific	medium
Experiment	content-independent	medium
Compartment	domain-specific	high
Species	domain-specific	high
Reaction	domain-specific	high
*Component	domain-specific	medium

Table 6.1.: Main meta-information stored in *mDB* (left column), their meta-information type (middle), and their importance for the ranking function (right column). Meta-information that are converted into a feature for the ranking are marked by an asterix (*). The notion of features and reasons for their associated values are discussed in detail in Section 7.3.1.

can participate in a reaction in the role of a reactant, modifier, or product.

Some representation formats (e. g. SBML) include the initial parametrization (parameter values to be used as the starting point for simulation) in the model description. For example, Figure 6.1 on page 125 shows the component `T0` and its value `0`. However, the initial values are not stored in the *mDB* representation. Instead, concrete values for model constituents should be stored separated from the model, for example using SED-ML (Section 5.4), or formats like SBRML (Section 3.1.1).

Model and structure meta-information Different kinds of meta-information are extracted from the original XML file, expanded and then stored. Table 6.1 lists the main meta-information. All information required by MIRIAM is covered; additional meta-information stored in *mDB* includes model versions, keywords, formalism, and a fine-grained storage of information about persons involved in the model evolution.

The meta-information covers information on the model level that refers to *content-independent meta-information* (e. g. the name of the model or the model creator). Additionally, a number of references to external resources can be stored to describe the modeled system. These meta-information refer to *content-dependent meta-*

information, in particular *domain-specific meta-data* (Section 2.3). Examples include the stored information on the model constituents, in particular the encoded species, reactions and compartments. Resources such as Gene Ontology, ChEBI, and KEGG were incorporated.

Meta-information that was considered relevant but did not have standardized encoding formats are the *simulation experiment related to a bio-model* and the *simulation algorithm* applicable to the model's simulation. Proposals for such formats have been developed and are described in Section 5.4.

6.3.3. Relation model

Figure 6.3 shows the full *mDB* relation model. The current design is closely related to the SBML schema, in the way that it has predefined component roles `species`, `reactions`, `compartment`, `parameter` and so on. Also, it stores `id` and `metaID` for each such component. If necessary, a generalization and relaxation of the schema to serve further formats will take place in the future. The mapping of other formats, such as CellML or πML is possible on the database structure.

In the following, the single *mDB* entities shown in Figure 6.3 are explained briefly[7].

model The `model` entity is one major entity of the *mDB*. It stores the `modelName`, declares versions as stable or unstable and keeps information on the latest version of a model. Each uploaded model gets its internal model ID by which it is identified within the system.

file The *file* entity contains information on the XML files associated with a model ID. The `modelFile` contains the XML code that is assigned to the model; optionally, the file location can be stored in the database. Some representation formats store their annotation information in a separate file which can be kept in the `anntotationFile` attribute. Each *mDB* model in addition has a versionFile associated that contains the version information, as described in Section 6.4.

modelAnnotation A number of meta-information needs to be stored with every uploaded model. The `modelAnnotation` relation stores all that information, including the model `creationDate` indicating when the model was originally created, the `modificationDate` indicating when the model was (last) modified, and the `submissionDate` indicating when the model was submitted to the

[7]A full description of the database schema, the entity and attribute structure is available from [Kolbe 2009].

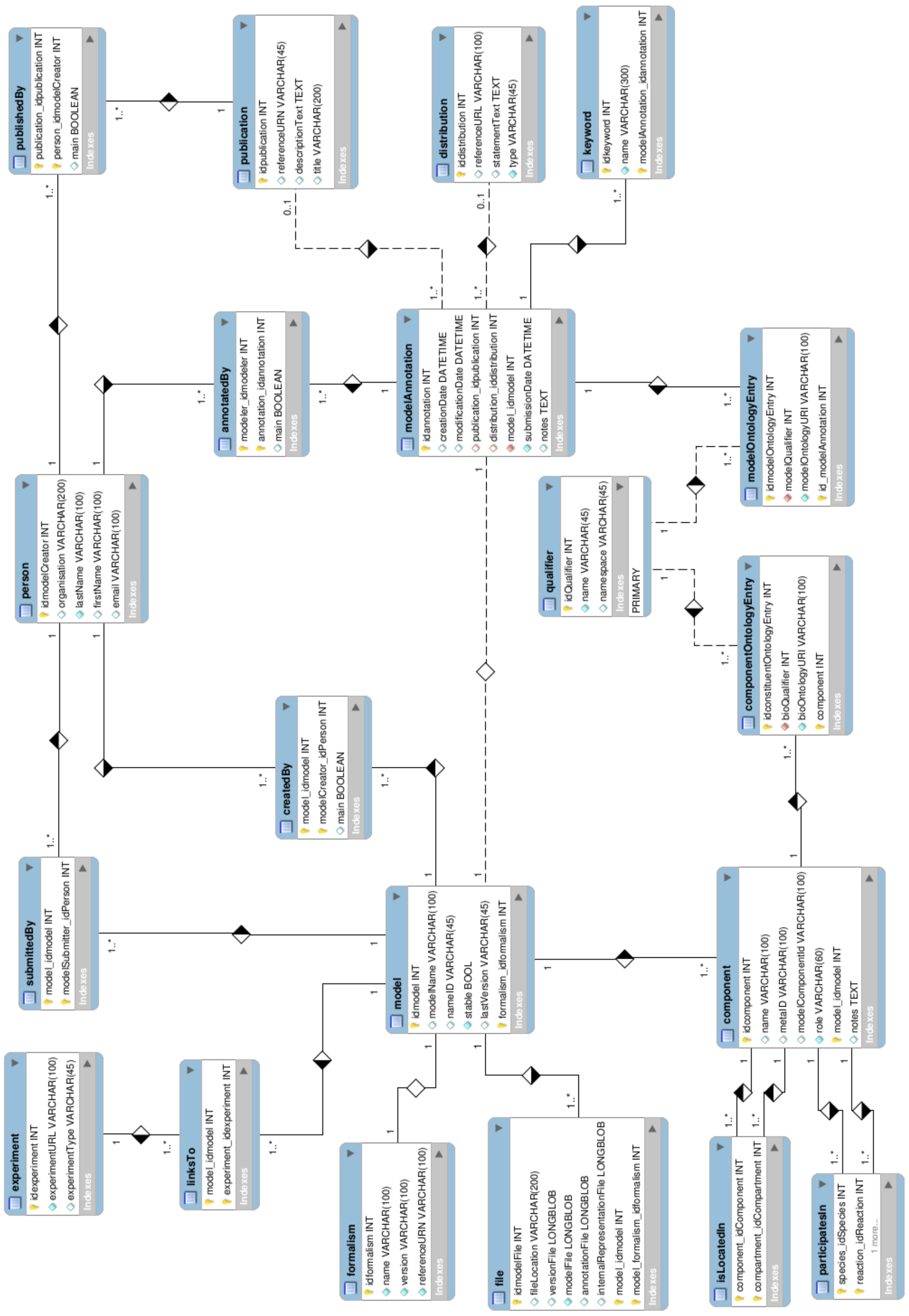


Figure 6.3.: The *mDB* relation model.

system. In addition, general notes, potentially stored with the model, are kept in the `notes` attribute.

The concepts behind the model file, the model annotations (`modelAnnotation` entity), and the model itself differ. Therefore, the relation model keeps them in three entities (although, from a database schema design point of view they could have been merged into one entity, due to the 1:1 relation between them).

A model is typically build on the basis of a reference publication. Information on the publication associated with a model is therefore stored in *mDB*. It is linked to a model via the model annotations.

publication The `publication` entity holds the `referenceURN` to the reference publication. In addition it stores the title of that publication, and a short description. The description typically corresponds to the paper abstract, as for example available from PubMed.

The life cycle of a model involves several different persons which are kept in *mDB* to comprehend who was involved in the model building process.

person The `person` entity contains general information about a person, including `firstName`, `lastName`, `organisation` and `email`. Each database entry has an ID. Depending on the relation a person occurs in, it can be a model submitter, creator, curator, or publisher.

submittedBy The relation associates a model with the persons who uploaded the model to the repository.

createdBy The relation associates a model with the persons who originally created the XML representation of the model.

annotatedBy The relation associates the model with the persons who annotated the model. Often, these correspond to the model curators as well.

publishedBy The relation associates the model with the authors of the reference publication. The additional `main` attribute indicates the main author.

A model can be linked to both, biology and simulation experiments; the links can be stored in *mDB*. Model-related information is considered valuable for later work on the model (e. g. to provide a user not only with the model code itself but also with instructions for simulation and with means to validate the model against real data).

experiment The `experiment` entity stores links to experiments associated with a model. The link, given in form of a URN (e. g. for SED-ML descriptions) or URL should point to an unambiguous, stable and valid third party resource providing the experiment description. The `experimentType` attribute determines whether the experiment is a biology or a simulation experiment.

linksTo The `linksTo` relation relates an experiment to a model entity.

Given the lack of standardized encoding formats for biology experiments, the `experiment` relation is solely used to link simulation experiment descriptions to a model ID.

Another type of relevant information stored explicitly for each model is the underlying model representation format.

formalism The `formalism` entity stores information on existing model representation formats: their `name`, `version`, and a `referenceURN` containing the definition of the representation format, in the ideal case a schema definition. The list may be extended if needed. Each entry in the `model` entity is associated with one entry in the `formalism` relation.

A bio-model contains a wide range of model annotations which are stored separately from the model but linked to a model ID. Examples are the publication, distribution statements, keywords and references to external resources which further describe the model.

distribution The distribution statement can optionally be stored for each model entity, either as free text (`statementText`) or via a link to an already defined distribution statement (`referenceURL`) such as the General Public License (GPL).

keyword `Keywords` are optional free text added by the model submitter, providing catchwords about it.

qualifier The qualifier relation stores pairs of qualifier `name` and `namespace`. Each defined qualifier has an ID that is referred to from the `modelQualifier` or the `bioQualifier` respectively. The relation currently stores the qualifiers available from the *biomodels.net initiative*. Further qualifiers may be added on demand.

modelOntologyEntry A model's meta-information is extracted when the model is imported into the database. Annotations that concern the model as a whole (model annotations) are stored in the `modelOntologyEntry` entity. The `modelOntologyURI` is kept together with a link to the pre-defined list of `modelQualifier` stored in the `qualifier` relation.

Besides storing the model and model meta-information, the different components forming a model are extracted and stored together with their annotations.

component The `component` entity is a generic concept capable of storing all kinds of annotated model constituents. A `name`, a `metaID`, and a `modelComponentID` (corresponding to the components internal model id) can be stored for each model constituent. A mandatory attribute of the component relation is the component `type` which is one of a predefined list of names (e. g. `species` or `parameter`. Additional notes can be stored for each component.

componentOntologyEntry Similarly to how the `modelOntologyEntry` stores meta-information on the model, the `constituentOntologyEntry` stores the constituents' meta-information. Each ontology entry is build of a `bioOntologyURI` and a `bioQualifier` putting the annotation in relation with the model constituent.

Examples for components relevant to SBML are the *species*, *reaction* or *compartment* constituents. Components can also map other representation format's concepts such as CellML "modules" or πML "definitions". Both can already be further characterized in *mDB* by the component's role.

Instead of storing the full meta-information in *mDB*, only the reference URNs are kept (`modelOntologyURI` and `componentOntologyURI`). The information encoded in the linked resource is incorporated in the indexing of the database (Chapter 7) and can be retrieved from that resource with available web services [Laibe and Le Novère 2007][8].

Further relations enable mapping the model structure on the database for later queries. They allow to re-create, for example, parts of the reaction network of a model directly from *mDB*. As a result, the XML model file does not have to be parsed with each structure query.

participatesIn If the component is a species, the information about the reaction it participates in is of interest for the network structure. The `participatesIn` relation stores exactly that information by relating two components at a time. Each relation has a specific `role` associated; predefined roles are `modifier`, `reactant`, and *product*.

[8]The more efficient way of resolving the URNs and storing the encoded information locally is *de jure* disputable and also leads to update problems. It has therefore been neglected.

locatedIn Often the location of a component is of interest, for example the location where a reaction takes place. The information is stored in the `locatedIn` relation which relates two components.

6.4. Model versioning

Current model repositories lack sufficient support of version control mechanisms. The CellML model repository, for example, provides different model versions on their web site but no information about the kinds of changes. The versioning information given by ModelDB is even more preliminary. While an SVN system is available for registered users of the curation system, BioModels Database currently does not provide general access to the different versions of a model. Changes cannot be tracked by the user and often they are not even mentioned on the web site. MIRIAM does not request to keep track on model evolutions. However, information on model updates should be required as it is of great interest for modelers working with a repository [Lloyd et al. 2008], for example, modifications in the mathematics or in the initial parameter assignments. If one wants to use a model that has been described in a paper and he retrieves it from a model repository, he should be made aware of the potential differences between the bio-model available from the repository and the bio-model description in the reference paper. Model changes are also relevant for meta-information based model retrieval (Chapter 7) as one needs to assure that the annotations of a model comply with the model semantics at the time it is queried (related to the *update problem* in DBMS). Furthermore, only explicitly defined model versions will allow for the usage of simulation descriptions such as SED-ML (Section 5.4).

Storing model changes The SBML modification history in principle allows to form a time line of modifications as a first step towards model versioning. The necessary information is implicitly given by the `dcterms:created` and `dcterms:modified` elements. However, to create the time line tree of a given model, one has to apply the following procedure:

1. Set date to model creation date.
2. Find the next later modification date.
3. Find all XML elements annotated with that modification date.
4. Create point on time line and list the modified elements.

5. Repeat 2-4.

The result is a time line showing the time points with modifications and to which model constituents those changes have been applied.

BioModels Database could make use of that information by providing the according XML SBML file for each of the modification dates, given a new file version is stored after each modification. The recreation process, however, is still time-consuming and does furthermore not provide a description of the type of changes made.

XML versioning approaches As mentioned in Section 3.3.1, existing model repositories internally use common version control systems such as *Subversion* or *Mercurial*. Those systems are specialized on the determination of diffs between text documents. In this work, a *diff* is considered an operation describing the transformation of a model version into its successor [Hälke et al. 2011]. The *Longest Common Subsequence* (LCS, [Hottinger and Meyer 2005]) algorithm is the standard procedure to detect changes. It detects differences between two files through line-by-line comparison of the string chains and identifies the longest sequence of common characters in two files. From that, a *diff file* is generated containing only the differences between the files. In addition, the operations necessary to transform the original document into a new version can be stored. However, those algorithms are not particularly suitable for XML documents [Hälke 2010]. For example, an SBML file does not consider the order of elements. Each tool importing and exporting an SBML file may use its own way of writing the XML without changing the meaning of its content. A tool might decide to write out the SBML file in one single line. The LCS algorithm does not detect a change in the XML accurately but returns the whole XML document as "changed" in the diff file. The problem of finding algorithms for the versioning of XML files in general has been studied in different work groups before [Cobena et al. 2002; Chen et al. 2004b]. Solutions for particular applications have been developed, including a version control for XML-encoded Office documents [Rönnau et al. 2005] or for pathway models in systems biology based on the definition of XML patches [Saffrey and Orton 2009]. A third interesting approach to XML-versioning with particular focus on XML native databases has just recently been published by Arévalo Rosado et al. [2009] who do not only store the difference between two documents but build a so-called *version_tree* which contains, for a given XML file, all modifications of all existing versions of that file. Hälke [2010] describes all mentioned approaches in detail. The conclusion of the evaluation is that bio-models demand an extended versioning technique that can, however, be based on existing solutions.

Identifying types of changes Particularly with the versioning of bio-models the *characterization* of applied changes is important. First of all, it will be interesting to know what exactly has changed in a model constituent by providing the *difference* (e. g. new and old parameter values). Secondly, it will be helpful to be informed about the *reasons* of change, as well as on the general *type* of change [Hälke et al. 2011]. This problem has to our knowledge so far not been addressed for the versioning of XML data – neither in systems biology nor in general. However, the complex nature of bio-models makes it an important task.

This section proposes an enhanced versioning system for bio-models. It is capable of identifying the difference between two bio-models, storing it and characterizing the changes. The approach can be used as an integrated extension to the *mDB*, but also with other model repositories.

The versioning concept is described in Section 6.4.2. Before that, the different notions of bio-model versioning and the question when to speak of a new model version are discussed in Section 6.4.1. A prototype implementation has been developed by Hälke [2010] and will be briefly described in Section 6.5.

6.4.1. Aspects of bio-model versioning

Depending on a bio-model's usage, different points of view on the important aspects of a model can be distinguished. A *modeler*, for example, might concentrate on the equations and mathematics that form the model, including the underlying laws that have been used to describe the system. A *biologist*, on the other hand, might prefer to view the model as an abstract thing that he can load into a simulation tool to gain knowledge about a system of interest. He will be particularly interested in the modeled biological entities, and their concentrations before and after running the simulation experiment. For verification, he might also consider the reference publication an important detail about a model. A *simulator* (the piece of software) will solely be interested in the underlying formalism (i. e. whether it can load the model), the initial parametrization of variables, and instructions how to simulate the given piece of code.

Each user group judges different changes in a bio-model relevant. For example, an update in the publication reference will be of high interest for the modeler and biologist, while it will be irrelevant for the simulator. A change in the annotation of a bio-model entity resulting in a reference to a different species will be highly relevant for a biologist, but again not at all matter to the simulator. A change in the underlying mathematics of the bio-model, however, is potentially not relevant to the biologist as long as he can see the result and understand the basic network structure, while it does matter to the modeler and the simulator.

The differing view points complicate the development of a generic versioning concept. When shall we speak of a new model version and when of a new model? In this work, each change leads to a new model version. Later on, tailored applications using the versioning approach might disregard certain changes as irrelevant. A discussion on how to classify the changes follows in Section 6.4.2.

Changes on the model source code A change in the XML file can either lead to a change in the bio-model structure, or to a change in the element/attribute names (according to the x-axis of the coordinate system shown in Figure 6.1 on page 125). A change in the XML tree could occur, for example, if a reaction is deleted from the reaction network or if additional species are introduced in the model. Changes in a string chain inside the XML file (e. g. changing the value or name of elements and attributes) are also relevant. The deletion of empty spaces and the reformatting of the XML representation, however, should be neglected by the system and not lead to a new model version.

There are border-line examples for the usefulness of introducing new versions for each renaming process such as changing the (string) name of an XML element. A change in a species name, for example from "fox" to "pickerel", might have consequences. A change in a species name is from "spec1" to "spec2", however, might or might not justify a new model version. Names like `spec1` and `spec2` are unlikely to hold any significant meaning. In opposition, a naming like "pickerel" in a predator-prey model might give a hint that the modeled system represents the entities in a predator-prey model in lakes. However, "pickerel" is only a string chain as well, and the true biological meaning of that entity should ideally be encoded in an annotation.

Changes on the annotation information A change in the annotation often leads to a change in the model's meaning. Annotations occur on all levels of a bio-model representation. Each update in an annotation is relevant and justifies a new model version.

6.4.2. Model versioning approach

Definition 6.2.1 defines a bio-model. The introduced solution distinguishes between versioning of the model source code m_S and the encoded annotation information m_A. Comments m_C in the XML code are so far neglected, however, one should be aware of their existence. The bio-model m_{bio} uses annotation information m_A to formally and explicitly encode biological aspects of the modeled system as precisely as possible. Including that information in the versioning process implicitly allows to infer on changes in the underlying biology.

The versioning approach stores the different bio-model versions during the model's life-cycle. Each version might be of later relevance and therefore is stored. Together with a model version, the changes with respect to the preceding version are identified and kept. A three-folded concept for such a versioning system, called *Biochemical Model Versioning System* (BiVeS[9]), has been developed by Hälke [2010] and is described in [Hälke et al. 2011]. It consists of an implemented *XML diff algorithm* that identifies the difference between two model versions through storage of the diff operations (insert, delete, update, move), an ontology for the classification of each diff operation, and a surrounding management system that keeps the versioning information in a so-called *model-history*.

XML diff algorithm One prerequisite for reliable versioning is a suitable algorithm to determine the differences between two instances of a bio-model. Hälke et al. [2011] propose an adapted version of the existing *XyDiff* algorithm[10]. While the original algorithm only supports the standard functions insert and delete, the new version also allows for the detection of moves, updates, and combinations of both [Hälke 2010]. At the same time, the algorithm's output is easy to interpret. The specifics of the implementation introduced by Hälke [2010] are the reduction of numbers of comparisons in the change detection, and the detection of moves. Section 6.5 on page 148 and Hälke [2010] provide more details on the implementation of the algorithm.

Ontology for change classification As mentioned earlier, changes on a bio-model may vary from simple reformatting of the XML code to updates on the model structure and annotations. These changes may or may not be relevant to a user, depending on the application. To overcome the problem of deciding for relevant and irrelevant changes, each identified change between two bio-model instances is annotated with a term from a change ontology [Hälke 2010] (Figure 6.4). The ontology has been developed in OWL and is divided into three main branches [Hälke et al. 2011]:

1. Terms from the *XML branch* reflect all changes on the technical level, which cover classical change operations (insert, delete, update, move) on XML elements m_S.

[9]`http://bives.sourceforge.net/`, last accessed 14 March 2011.
[10]`http://leo.saclay.inria.fr/software/XyDiff/`, last accessed 14 March 2011.

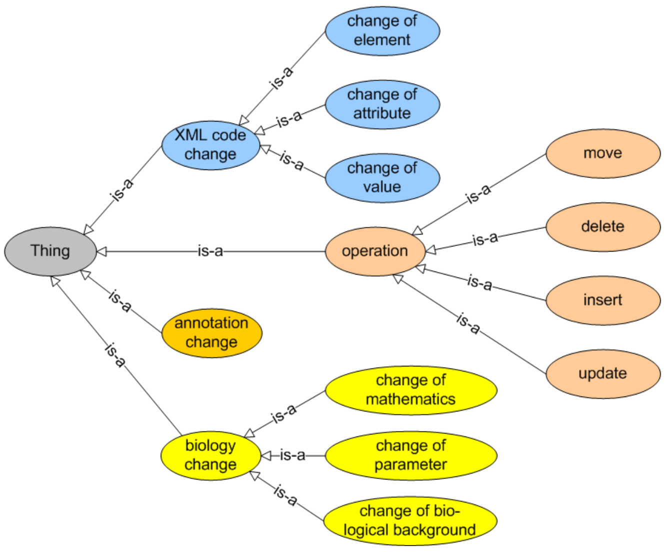


Figure 6.4.: First draft of an ontology for change classification in bio-model versions (skeleton). Adjusted from [Hälke 2010].

2. Terms from the *Annotation branch* reflect updates on the model annotations. The branch allows to separate simple XML code updates from updates on the annotations m_A.

3. Terms from the *Biology branch* reflect all changes on the modeled biological system such as changes in the parametrization, changes in the models background, or changes in the mathematical equations.

Version management system BiVeS manages the different bio-model versions and combines the generated diffs with the classifications given by the ontology.

Based on the ideas published in [Arévalo Rosado et al. 2009], the system only stores the differences between two bio-models, their hierarchical version, and branch information in a so-called *model history*. The model history consists of two parts, namely the *versionTree* which contains information on all versions of the model and the *diffList* which contains the set of diffs, each associated to a specific version [Hälke et al. 2011]. Diffs consist of change operations on the models, expressing in which version the corresponding component has been added, deleted or modified.

Each operation in the *diffList* can be annotated with terms from the different ontology branches. Annotations from the XML branch, characterizing the kind of change on the technical level, can be assigned automatically. Changes in annotations (Annotation branch) can in many cases be detected, while terms specifying changes on the biology have to be manually associated with the corresponding diffs. Besides annotations from the ontology, additional information on each branch (e. g. names, change submitter, or comments) can be stored.

6.5. Implementation

πML

To enable the exchange, RDF-based annotation and a better model reuse for π Calculus models, the πML format has been developed. It is an XML-based format for the encoding of basic, stochastic and polyadic π Calculus models.

The language is encoded in XML schema. The core πML schema definition, the extensions, and sample models are available from `http://piml.svn.sourceforge.net/`. A library for the creation of πML representations out of software tools does not yet exist.

mDB

The relation model introduced in Section 6.3.3 has been implemented in a MySQL database. However, the relation model design allows for easy implementation of the developed database schema using other DBMS.

tinyParser The *tinyParser* developed by Robert Hälke during his study thesis has been extended to import SBML models into *mDB*. The import module is part of the test framework for ranked model retrieval. Details are given in Section 7.5.

Versioning with BiVeS

The versioning concept has been extended and implemented by Robert Hälke during his Diploma Thesis [Hälke 2010]. Existing approaches to the determination of differences between documents in general and XML-files in particular have been evaluated and tested for the bio-model versioning. However, traditional approaches such as SVN, have not been judged satisfying. The developed library for versioning of bio-models, called *The Biochemical Model Versioning System* (BiVeS, [Hälke et al. 2011]), is a Java-based implementation of a versioning system that is build of the three different components introduced in Section 6.4.2.

XML diff algorithm The algorithm for the determination of diffs in XML documents, called *XyDiff*, is the basis for the developed *XML diff algorithm* used in BiVeS.

Operation	Path	Hash value
move (exact match)	$p(i) \neq p(i')$	$h(i) = h(i')$
move (partial match)	$p(i) \neq p(i')$	$h(i) \neq h(i')$
update	$p(i) = p(i') \vee p(i) \neq p(i')$	$h(i) \neq h(i')$
insert	$i'_n \notin M' \wedge i_n \in M$	$i' \notin M' \wedge i_n \in M$
delete	$i_n \notin M \wedge i'_n \in M'$	$i \notin M \wedge i'_n \in M'$

Table 6.2.: Identifyable operations based on the changes in the path and hash values of the combined identifiers Hälke [2010]

Each node in a bio-model's XML tree has a unique identifier generated; the identifier consists of a node's XPath p and its hash value h. A comparison of two bio-model file versions *doc* and *doc'* can be reduced to a comparison of both documents' sets of identifier M and M'. In a first step, all identical identifiers are removed from both sets and only changed identifiers are considered. Depending on the kind of difference between two nodes in their path and/or hash values, the type of change can be determined being either a move, an update, an insert or a delete. To do so, the remaining identifiers $i \in M$ and $i' \in M'$ are compared and the changes are classified as shown in Table 6.2 (taken from Hälke [2010]).

Moves can easily be detected as they only change the p-part of the identifier and keep the h-part. The full algorithm is explained in [Hälke 2010, pp. 45-48]; the abstract XML diff algorithm is shown in Figure 6.5 (taken from Hälke [2010]).

Ontology for change classification The three-folded ontology encoding the different kinds of changes on a bio-model has been implemented in the Web Ontology Language (OWL), using the software tool *Protégé*. The ontology is in a preliminary state. The XML branch describing the possible XML changes (update, insert, delete of elements, attributes or values) has been fully modeled. However, the other two branches are still skeletons. The Annotation branch contains classes that describe the different possible annotation updates. Currently changes in the annotation can be marked as new annotations (insert), updated annotations (update), or removed annotations (delete). The most preliminary part of the ontology are the biology-related classes. Only the three different classes parameters, mathematics, and background have been defined so far, denoting the change of model parametrization, changes in the mathematical encoding of the model, and changes in the biological background, respectively.

The assignment of ontology entries to a particular diff is therefore either automatic (for XML changes), semi-automatic (for annotations) or user-driven (for the biology branch).

```
{First parsing of document, create sets of identifiers}
    for each node in $doc$ do
        generate $p(i)$
        generate $h(i)$
        insert $i$ into $M$
    end for
    for each node in $doc'$ do
        generate $p(i')$
        generate $h(i')$
        insert $i'$ into $M'$
    end for
{Insert all unequal identifiers into the sets $M_{diff}$ and $M'_{diff}$
    for each $i \in M$ and $i' \in M'$ do
        if $p(i) \neq p(i')$ or $h(i) \neq h(i')$
            insert $i$ into $M_{diff}$
            insert $i'$ into $M'_{diff}$
        end if
    end for
{Second parsing of documents, detect all changes}
    for each $i \in M_{diff}$ do
        find moves
        find updates
        find inserts
        find deletes
    end for
```

Figure 6.5.: Abstract XML diff algorithm as developed by [Hälke 2010].

Version management system The BiVeS library manages the storage of identified diffs in a `diffList` and the version history in terms of branches and merges in a `verionTree`. Together those two elements represent the so called `modelHistory`. The output is an XML format complying with the XML Schema definition available from `http://bives.svn.sourceforge.net/`. The `modelHistory` stores all diffs for a given model in a single file, including the history and optional `metaData`.

6.6. Summary

Results

This chapter introduced a format-independent storage and versioning approach for XML-encoded, annotated model representations of bio-models. A database schema was proposed to realize the storage approach. It has been implemented in a MySQL database (*mDB*) as part of a test framework for bio-model retrieval (*Sombi*). Furthermore, a versioning concept for models stored in *mDB* has been developed. As a proof of concept, models in SBML format have been imported in a test implementation, stored and versioned with the given concepts. How the format-independent storage works for other formats has been shown conceptually, using the example of a developed XML format for π Calculus models.

The proposed storage system differs from existing repositories by a fine-grained storage of meta-information extracted from the model representations and stored in a relational schema. Consequently, the advantages of DBMS can be used, including a thorough user rights management, transaction support, and data consistency. The database design allows to store both, data and information about models (encoding mainly in the model representation format) and model meta-information (encoded mainly in the annotations). The meta-information storage is format-independent. Moreover, *mDB* is coupled with further databases providing information on related experiments and results. The integration of all three sources allows for queries not only addressing one step in the modeling and simulation workflow, but to integrate the constraints on different steps. For example, a user can restrict his search results to models for which valid experiments exist.

Discussion

mDB vs UCL Beacon The *mDB* structure elements introduced in Section 6.3.3 can be mapped on the meta-model developed by the UCL Beacon project (Section 3.3.1). The mappings are shown in Table 6.3. The focus of the UCL Beacon project is on the description of the model surrounding. *mDB*, on the contrary, delegates

Concept	***mDB***	**Beacon**
model definition	model	model
concise description of modeled system	modelAnnotation	aspect
values used to instantiate the model	experiment (*implicit*)	context
cause that led to model construction	-	assumption
information sources	publication	origin
experiment description	experiment	-
execution information	experiment (*implicit*)	engine
results and conclusions	experiment (*implicit*)	interpretation
simulation/biology experiment result	experiment (*implicit*)	-
persons involved in M&S process	person (*only modeling*)	person

Table 6.3.: Table-to-table mapping of the *mDB* and the UCL Beacon meta-model regarding selective concepts. *Implicit* entities denote that the concept is implicitly contained in the relation model and can be derived (e. g. by resolving a reference to another data resource such as *sDB* (Section 5.4.3)).

the description to the related experiment description format (SED-ML), using a single `experiment` entity (Figure 6.3 on page 138). The focus of *mDB* lies on the detailed storage of ontological information about the model and its constituents (i. e. `modelOntologyEntry`, `componentOntologyEntry`). Further focus lies on the storage of different model versions. Both aspects are disregarded by the UCL Beacon project.

Coupling the versioning system with existing storage solutions The first attempt for versioning was directly integrated in the *mDB* schema through providing predecessor and successor information for each model and each model annotation. The current schema, however, handles versioning externally. A model in *mDB* only contains a link to the last version of its kind. The remaining versioning information is managed by an external tool that creates and updates a version history for each submitted model.

The developed system can therefore be coupled with existing model repository solutions: For each new upload of a model, the system can determine the change with respect to the model's previous version. As the versioning is done outside the bio-model files, it does neither affect the bio-model code, nor the storage solution. It will be useful, however, to extend the upload interface to provide the user means for (semi-automatically) annotating the identified changes with terms from the change ontology. A sample architecture for the integration of BiVeS with existing SVN repositories has been proposed by Hälke/Henkel during the *3rd model meeting*[11].

[11]`http://dbis.informatik.uni-rostock.de/~dk103/files/bivesIntegrationSvn.pdf`, last ac-

To reconstruct the history of a bio-model, its associated modelHistory file needs to be read. All existing versions can be gained easily from the `versionTree`. The diffs leading from one version to the other can be obtained from the `diffList`.

Depending on the application of the versioning tool, different modifications may be shown to and respectively hidden from the user. The concept of change annotation allows for the definition of rules for the diff presentation to the user by evaluating the ontology terms associated with each of the diff operation.

Generic versioning concept The current versioning prototype concentrates on the versioning of SBML models. However, a more generic solution to the problem is needed to enable the versioning for all different kinds of model encoding stored in *mDB*. The solution introduced in [Hälke 2010] has to be tested for applicability to CellML and πML. Once it is successfully integrated, it has to be evaluated regarding both, usability and performance.

Relating simulation data and model data The internal ID is used to address a particular experiment from outside, for example to link to it from the `experimentURL` attribute in the `experiment` relation in *mDB* (Section 6.3.3 on page 138). Alternatively, if both databases are integrated, then the `experimentURL` attribute in *mDB* contains the `idSimEx`.

Relating experimental data and model data The incorporation of experimental data in the ranking process is a next step towards enhanced model retrieval. Experimental data therefore has to be stored in a way that it can be integrated with the current model storage system. Starting from the year 2006, we have developed the *experimental database*[12] (*eDB*) to provide a simple and tailored solution for the storage of biology experimental data collected in the dIEM oSiRiS project. *eDB* stores Western blot and microscopy experiment setups, but also the result data. It is implemented as a MySQL database with a PHP frontend, supporting the definition of new experiments and data upload for each experiment as well as providing sophisticated user right management for sharing experiments with collaborators.

The collected data needs to be annotated (preferably using MIRIAM URNs) to be incorporated in the retrieval process. This has to date not been done, one of the major issues being the lack of experts for the data annotation process. The OBI foundry investigated enhanced annotation of experiment descriptions using ontology references [Brinkman et al. 2010]. One step towards solving the biology experiment

cessed 17 December 2010.

[12] `http://diem.informatik.uni-rostock.de/`, last accessed 12 October 2010.

description problem in a more general sense is the integrated maintenance of models, simulations and result data in a common workflow, as done by the *FuGe* project (`http://fuge.sourceforge.net/`); it suggests a modular approach to describing, storing and exchanging complete biological workflows. Once annotations exist in *eDB*, the retrieval concept can be extended to include also data associated with a model.

Linking models, simulations, and data with SED-ML The information provided by the different developed databases (*mDB*, *eDB*, *sDB*) enables integrative search for models with associated simulation and biology experiments. Going from biological data to the modelling approach and the subsequent simulation task (Figure 1.1 on page 3) demands the integration of the corresponding data. Specific problems that arise have been characterized by Unger and Schumann [2009] in the context of visualization research. They define three so-called "process levels": the model level, the experiment level, and the level of multi-run simulation data[13]. On a given model, several experiments may be performed. Each experiment may have a number of different simulation data sets associated with it. The model level describes the model structure, including its components (chemical compounds) and reactions. The experiment level describes the specific settings of a model for which the simulation is performed. Unger and Schumann [2009] thus consider an experiment an "instance of the model under certain conditions". The multi-run simulation data level finally describes the system state over time which is determined by the values of the state variables (of the discrete event based system) over time[14]. The simulation data may vary even for a single experiment (instance of the model). This is due to the stochasticity in the system. The connectivity of the different levels is essential for the Mosan concept: The model level – in addition to the model structure – links to the simulation data distinguished for the different simulation experiments. Furthermore, the analysis of an experiment needs to work on the links to the multi-run simulation data.

SED-ML (Section 5.4) can help to encode the linking between bio-models and result data, as well as simulation and experimental data. The latter step is needed in order to validate the correctness of the model. A sample SED-ML file integrating the different information sources is shown in Listing 6.3. As SED-ML lacks elements for the encoding of linked result data and experimental wet lab setups, the information

[13]The framework is defined for stochastic simulation of biochemical reaction networks as discrete-event systems" [Unger and Schumann 2009], but it could be extended towards other modeling approaches.

[14]Please note, that in the scope of this work the term "result set" is used rather than "simulation data".

is stored as annotations, using the `<notes>` element.

```
1  <sedML>
2   <listOfSimulations>
3    <uniformTimeCourse id="simulation1"  initialTime="0" outputStartTime="0
        " outputEndTime="1000" numberOfPoints="1000" >
4     <algorithm kisaoID="KiSAO:0000088"/>
5    </uniformTimeCourse>
6   </listOfSimulations>
7   <listOfModels>
8    <model id="model1" name="Repressilator" language="SBML" source="
        mdb:0001" />
9   </listOfModels>
10  <listOfTasks>
11   <task id="task1" name="Oscillation using a deterministic simulator"
        modelReference="model1" simulationReference="simulation1">
12    <notes>
13     <experimentSetup name="Western blot experiment setup"
14     experimentReference="edb:056209" />
15    </notes>
16   </task>
17  </listOfTasks>
18  <listOfDataGenerators>
19   <dataGenerator id="time" name="Time" [..] />
20   <dataGenerator id="LaCI" name="LaCI repressor" [..] />
21   <dataGenerator id="TetR" name="TetR repressor" [..] />
22   <dataGenerator id="CI" name="CI repressor" [..] />
23  </listOfDataGenerators>
24  <listOfOutputs>
25   <plot2D id="plot1" name="protein numbers per time point">
26    <listOfCurves>
27     <curve [..] xDataReference="time" yDataReference="LaCI" />
28     <curve [..] xDataReference="time" yDataReference="TetR" />
29     <curve [..] xDataReference="time" yDataReference="CI" />
30    </listOfCurves>
31    <notes>
32     <simulationResult name="result time course simulation in Copasi"
33     resultReference="http://www.comp-sys-bio.org/static/examples/example2
          .xml" />
34    </notes>
35   </plot2D>
36  </listOfOutputs>
37 </sedML>
```

Listing 6.3: Prototypic SED-ML file specification for integration of model, biology experiment, and simulation information for Mosan. Namespaces omitted, no complete definition of all necessary elements.

The SED-ML file specifies the model to load into Mosan, namely the model in *mDB* with ID `0001` (l. 8). The model is then used in the task definition (ll. 11-16) together with the defined simulation setup (ll. 3-5). That specific task corresponds to the biological experiment with ID `056209` as described in *eDB*. The chosen output of the experiment is a 2D plot that shows the change in concentration of `LaCI` (l. 27),

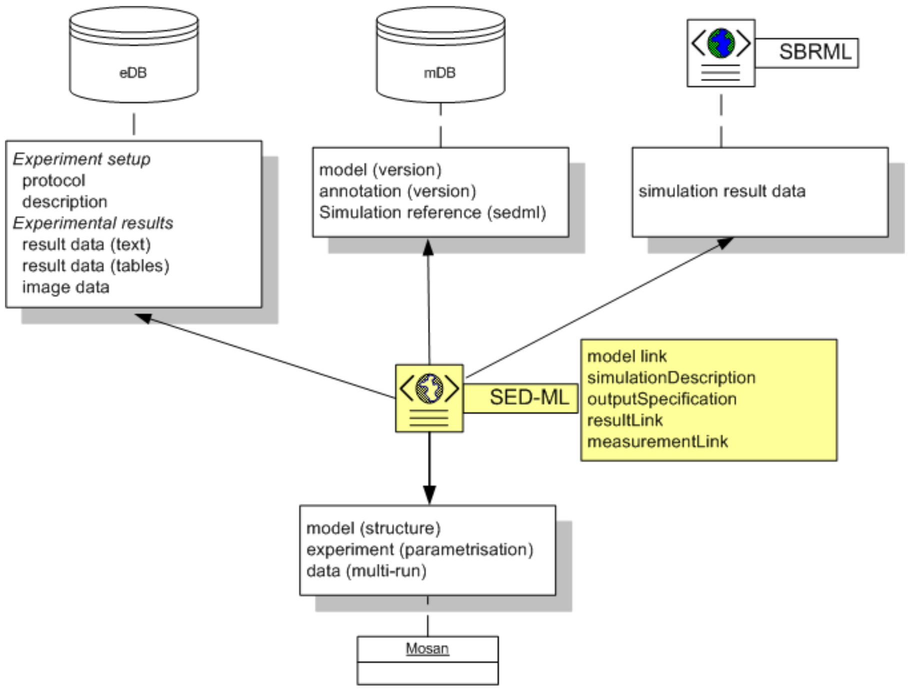


Figure 6.6.: Conceptual design: SED-ML for the combined storage of models, related simualtion setups and simulation results in Mosan

`TetR` (l. 28) and `CI` (l. 29) over time. The result data produced by the output is exported and stored in an SBRML file which is referred to in the `notes` element (ll. 31-34).

SED-ML has been discussed as the exchange format to provide unambiguous linking between the three information sources in Mosan (Figure 6.6). *mDB* represents the model resource; *eDB* is the source for experimental data, and SBRML encodes the simulation result data. A SED-ML description then links one or more instances of a model from *mDB*, a corresponding simulation description stored in *sDB*, the result data set for the experiment in SBRML format (stored in an online resource, or in *mDB*), and finally the reference wet lab experiment (stored in *eDB*). For the application on Mosan, all information is then loaded into the tool via a defined interface. The information encoded in SED-ML is mapped on the Mosan object structure (Table A.12 in Appendix A.5.4) and then visualized.

Model coupling One of the challenges of systems biology is the construction of complex models from smaller existing models [Liebermeister 2007]. The construction of large-scale bio-models can be supported by methods for successful integration of smaller models.

Li et al. [2010] mention several problems: Models from the same domain space can still show very different behavior and as such exhibit significant variations. Models are developed by different researchers, at different times, with different background knowledge and a different perspective on the problem. They also exist in different formats. All the reasons hamper the model reuse in a coupling scenario.

However, cells themselves provide an obvious and natural form of biological modularity – by physically partitioning off biochemical reactions [Kitano 2002b]. The investigation in modularization and then coupling of model components (including full models) seems promising. Spiegel et al. [2005] did a case study on contextual information necessary to reuse and compose simulation models. This study should be evaluated, and its application on *mDB* should be tested. *mDB* provides a very modular storage solution for different model components. Using the annotations, components with different names can be compared with regard to their biological function or meaning. The idea of integrating different models based on their ontological information has already been presented in [Gennari et al. 2008] with a focus on multi-scale simulation models. The approach uses its own ontology. The ideas have to be evaluated regarding the storage scenario in *mDB*.

Once matching models are identified, the approach to composing simulation models based on an XML-description of their interfaces [Röhl 2008] might enable biomodel coupling. In a first step, the meta-information based retrieval (Chapter 7) can be used to find relevant model. In a second step, information on the model interface as encoded by Röhl [2008] could be incorporated to check for composability of the models.

7. Meta-information-based ranked retrieval

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 Stoker's Dracula, 1897)

Soldatova and King [2005] state that "more and more biological data are stored on computers, the problem of efficient retrieval and analysis of these data becomes the most important scientific bottleneck, and the problem is particularly acute in biology because biological data are notorious for their complex form and semantics". The provision of meta-information for the determination of a model's *meaning* as discussed in the previous chapters is the first step towards model reuse. The process of *retrieving* the given information and making sense of it is at least as important.

The following chapter describes the application of IR techniques on the retrieval of annotated bio-models. Section 7.1 discusses the problem in detail, followed by the prerequisites for the ranking system (Section 7.2). Section 7.3 then explains the concept of *bio-model retrieval*, including an introduction to the different feature types considered in the ranking process. To help software developers evaluating available ranking functions for their data base, I propose a modular test framework called *Sombi* (Section 7.4). The chapter continues with an overview of implementations (Section 7.5) and then closes with a summary of results and a short discussion (Section 7.6).

This work showed the successful application of IR techniques on bio-model retrieval [Köhn et al. 2009; Waltemath et al. 2011c]. The developed concept has been implemented to support ranked retrieval in BioModels Database [Henkel et al. 2010].

7.1. Problem statement

Ongoing efforts to foster reuse in CSB consider artifacts representing full models and artifacts representing model components. For example, Krause et al. [2009] investigate improved model merge capabilities based on overlapping model constituents.

Klipp and Liebermeister [2006] furthermore introduce a set of so-called "recurrent building blocks" in the modeling of signaling pathways, discussing a set of sample models of particular signaling pathways, including Jak-Stat or the WNT/β-catenin pathway. That set of models, although showing different designs, reuses the formerly introduced building blocks. Unfortunately, the identified building blocks are not made available for modeling in an open component database. As another example, the CellML community proposes a "Repository of modular Modeling Components for Synthetic Biology" [Cooling et al. 2010] to enable modular biological modeling [Cooling et al. 2008]. A common limitation of the abovementioned approaches is the assumption that the models are already known to be suitable for reuse. The preceding step of *retrieving* relevant model constituents is neglected.

Contrary to semi-structured XML documents considered in literature, bio-model encodings do not contain all relevant information for the search. More than that, the documents contain pointers to external resources which then provide the corresponding information. Figure A.6 on page 200 gives an example: The information encoded directly in the SBML file is limited to the species name `E` (a string) and the ID `species_1`. Linked to that species is an annotation, pointing to the external resource at `urn:miriam:uniprot:P00439`. The location of the species is given as `compartment_1`. To incorporate the knowledge provided by the URNs, external resources have to be resolved and processed.

Existing search facilities in model repositories are neither sufficient, nor detailed enough. Query possibilities are limited and not all information available from the model's structure and its annotation are used to find relevant models for a query. A ranking is missing, and the user needs to be informed about the reasons why a particular model has been returned for his query. Also, flexible ways of adapting the search to a user's demands are required.

Bio-model retrieval necessitates several different *types* of queries: Some preliminary information is gained from the XML code itself. The structure information gives insights on the system's underlying network structure. Finally, annotations, provided as links to external resources, help to identify the nature of the bio-model.

7.2. Prerequisites

The *similarity* between two entities (models, model constituents, queries) A and B is related to their commonality. The more commonality they share, the more similar they are [Lin 1998, intuition 1].

We adapt the general definition for reuse in the context of Information Systems Development [Brash 2001] (Section 2.5) to the more restricted application area of *bio-model reuse*:

Definition 7.2.1 (Bio-model reuse)**.** Bio-model reuse *is the employment of a previously used and explicitly defined bio-model artifact in another bio-model development process.*

We furthermore define a bio-model retrieval model for the determination of similarities between a query and a set of bio-models, based on the definition for information retrieval [Baeza-Yates and Ribeiro-Neto 1999]:

Definition 7.2.2 (Bio-model retrieval model, adapted from [Henkel et al. 2010])**.** *A bio-model retrieval model is a quadruple* $(M, Q, F, R(q_i, m_j))$ *where*

1. *M is a set of representations m_i for bio-models in the collection*
2. *Q is a set of queries q_i, each represented as a set of triples $< f, t, w >$, where f is a feature, t is a term and w is a weight*
3. *F is a computational framework for modeling bio-model representations, queries and their relationships*
4. *$R(q_i, m_j)$ is a ranking function defining ordering among the models m_j with regard to the query q_i*

7.3. Ranked retrieval of computational biology models

Progress comes when what is actually true can be separated from what is only believed to be true.

(Basili et. al)

The following section discusses the conceptual architecture for ranked bio-model retrieval. The features incorporated in the retrieval and ranking are categorized (Section 7.3.1) and then explained (Section 7.3.2). The support for structure queries is discussed in Section 7.3.3. Afterward, the whole architecture is described in detail (Section 7.3.4) and a sample query is given (Section 7.3.5).

7.3.1. Feature dimensions

Available meta-information for bio-models as described in the different standardization efforts and formats shown in Figure 3.1 on page 44 have influenced the feature dimensions (i. e. MIRIAM and its external resources, SBO, SED-ML and KiSAO,

dim	Name	Weight	Sample features
1	`Administrative`	very low	file name, version, formalism
2	`Persons`	medium	author, encoder, submitter
3	`Publications`	high	title, abstract, full-text, journal
4	`User generated content`	very high	keywords, remarks, tags
5	`Dates`	low	submission or modification date
6	`Constituents`	very high	reactions, species, functions
7	`Experiment`	high	behavior, simulationAlgorithm

Table 7.1.: Identified dimensions. Extended from [Henkel et al. 2011] with an additional seventh dimension "Experiment".

and TEDDY). In addition, features have been derived from the representation format structures. Further features have been added which can be extracted from the information available during model submission to a model repository. Their values originate from queries on the relation model shown in Figure 6.3 on page 138 (implemented in the *mDB*).

From the study of available meta-information, seven *dimensions of features* are distinguished (Table 7.1, following [Henkel et al. 2011]).

Administrative data contains specific information about the model file and representation format used to encode the model. It holds information suitable to filter by different model attributes (e. g. the formalism a model is encoded in, or its version).

Persons contains meta-data about persons related to the model. Each person has a designated role (i. e. author, encoder, submitter of a model, or author of the reference publication of a model).

Publications contains information about publications related to a model, including reference descriptions, reviews of a model, or related dry and wet lab results.

User generated content contains information provided by a user (e. g. keywords, tags or remarks). This information reflects the users point-of-view on a biomodel.

Dates contains different timestamps that enable time-wise filtering of models (e. g. creation date, modification date or submission date).

Constituents contains information about encoded model constituents, both directly encoded in the model, and provided via annotations.

Experiment contains information about experiments related to a model. Those can be either simulation experiments or Wet Lab experiments, and their associated information.

The *experiment* dimension contains result data, but also experiment descriptions, for example encoded in standardized XML-format or in described in free-text. The information is different to the reference publication, which is a long textual document or report describing the system under study, the experimental setup and the experimental results.

All but the *constituents* dimension can be applied to the various modeling formalisms implementing the MIRIAM specification. They are format-independent. The *constituents* dimension, however, is format-dependent. A format intended for neuronal modeling will certainly have a different constituent spectrum than a format for kinetic rate law models. This does not touch the concept of annotation though. Detailed descriptions of the different dimensions are given in Section A.5.2.

Relation to the coordinate system

The coordinate system in Figure 6.1 on page 125 already introduced different kinds of searchable entities. Its axes can be assigned to the aforementioned dimensions: From the data level (data axis), information on the model parametrization (*experiment dimension*) can be inferred, for example "find me a model where the K_M value of the `KM` parameter is `40`".

The constituent names (*constituent dimension*) are also part of the data level; due to the fact that they are often rather meaningless, they are not considered of high priority for the search. From the meta-model, information on the biological meaning of the model constituents can be inferred (data level), as well as information on the mathematics used to encode the model (structure level), for example, "find me models encoding phosphorylations". Both kinds of information are part of the *constituents dimension.*

Model information (*administrative dimension*), publication information (*publication dimension*), dates (*dates dimension*) and the user generated content (*user generated content dimension*) are not covered by the coordinate system. They either belong to the model level (instead of annotating model entities), or they belong to the data provided by the DBMS and manually by the user, respectively.

7.3.2. Features (conceptual)

The conceptual design of the ranked retrieval approach builds on a variety of different features for the model similarity determination. Regarding the classification given

feature name	meaning	origin
ID	internal modelID	modelID (*mDB*)
additionalID	modelID in the XML file	repFormat / *mDB*
path	model file location	system
content	aggregated content of all feature values	system
formalism	representation format	formalism (*mDB*)

Table 7.2.: Features in the administrative data dimension with the feature name, its interpretation (meaning), and the origin of information.

in Section 2.3.1, they belong to the class of associated content-independent and content-dependent meta-information. All conceptually identified features can either be extracted from the model representation format during model submission, or from the data stored inside the database *mDB*. The following section introduces all conceptually identified features while the subset of features used in the prototype implementation in BioModels Database is provided in Section 7.5.

Features of the administrative data dimension

The features of the administrative data dimension and where they are obtained from are shown in Table 7.2. The administrative data dimension keeps the information on the internal model IDs in both the database system (`ID`) and the model file itself (`additionalID`). The model location is furthermore indexed (`path`), and the particular encoding format (i. e. `formalism`) is indexed. The `formalism` allows to restrict a search to particular encodings. For example, if the user is depending on the use of a particular simulation environment, the encoding of the model must be so that he will be able to run the model.

Finally, the administrative data dimension indexes all other feature values of all dimensions in the `content` feature to allow for a quick retrieval based on general queries which do not address a particular feature [Henkel 2009].

Features of the persons dimension

The features of the persons dimension and where they are obtained from are shown in Table 7.3. The ranked retrieval uses a fine-grained splitting of person roles involved in the model development process. Different features distinguish between the model submitter, creator, publisher and curator. For each role (except the submitter to the repository who is a single person), the main person is furthermore determined. Explanations of the different person roles have been given in the *mDB* relation model description in Section 6.3.3 on page 137.

feature name	**meaning**	**origin**
submitter	model submitter (repository)	submittedBy (*mDB*)
creator	model creator	createdBy (*mDB*)
publisher	publisher of reference publication	publishedBy (*mDB*)
curator	person who curated the model	addedBy (*mDB*)
mainCreator	main creator (model file)	createdBy:main (*mDB*)
mainPublisher	main publisher	publishedBy:main (*mDB*)
mainCurator	main curator (model file)	annotatedBy:main (*mDB*)

Table 7.3.: Features in the persons dimension with the feature name, its interpretation (meaning), and the origin of information.

feature name	**meaning**	**origin**
submissionDate	date of submission to repository	submissionDate (*mDB*)
creationDate	date of model creation	creationDate (*mDB*)
modificationDate	date of last model modification	repFormat / versioning

Table 7.4.: Features in the dates dimension with the feature name, its interpretation (meaning), and the origin of information.

Features of the dates dimension

The features of the dates dimension and where they are obtained from are shown in Table 7.4. The dates dimension consists of three features: the date of first submission of a model to the repository (`submissionDate`), the date of model creation (`creationDate`), and the date of last modification of a model (`modificationDate`).

Features of the publication dimension

The features of the publication dimension and where they are obtained from are shown in Table 7.5. The features in the publication dimension comprise information extracted from the reference publication, including the publication identifier,

feature name	**meaning**	**origin**
publicationID	publication identifier	referenceURN (*mDB*)
publicationText	publication abstract	descriptionText (*mDB*)
publicationTitle	title (reference publication)	title (*mDB*)
distributionStatement	distribution type (model)	distribution:type (*mDB*)

Table 7.5.: Features in the publication dimension with the feature name, its interpretation (meaning), and the origin of information.

feature name	**meaning**	**origin (*mDB* and repFormat)**
modelName	model name	repFormat/model:name
species	species name	component:name[@role=species]
reaction	reaction name	component:name[@role=reaction]
compartment	compartment name	component:name [@role=compartment]
parameter	parameter name	component:name[@role=parameter]
function	function name	component:name[@role=function]
event	event name	component:name[@role=event]
modelURI	model URI	repFormat/modelOntologyURI
speciesURI	species URI	repFormat/componentOntologyURI
reactionURI	reaction URI	repFormat/componentOntologyURI
compartmentURI	compartment URI	repFormat/componentOntologyURI
parameterURI	parameter URI	repFormat/componentOntologyURI
functionURI	function URI	repFormat/componentOntologyURI
eventURI	event URI	repFormat/componentOntologyURI

Table 7.6.: Features in the constituent dimension with the feature name, its interpretation (meaning), and the origin of information.

typically a URN pointing to a PubMed entry.

So-called model-related meta-information encoded in the annotations include information on the model that is only derivable by querying third-party resources, such as controlled vocabulary, or ontologies. Features include the `referenceURN` and the `title` of the reference publication, the `publisher` (refer to the `publishedBy` relation in the person dimension in Table 7.3). In addition, the textual description of the linked publication (as provided by external resources) is parsed and indexed, building the `publicationText` feature. The title of the publication is indexed separately. Finally, the `distributionStatement` makes up another feature in the publication dimension.

Features of the constituents dimension

The features of the constituents dimension and where they are obtained from are shown in Table 7.6. The constituents dimension is split into features regarding the name element contents in the model file and features regarding the annotation contents in the model file.

Model-related meta-information in the constituent dimension includes the different URIs given for the model itself and the model constituents. The index is build

feature name	meaning	origin
keywords	tags (model submission)	keyword:name
modelDescription	model description	repFormat/ modelAnnotation:notes
componentDescription	component description	repFormat/ component:notes

Table 7.7.: Features in the user generated content dimension with the feature name, its interpretation (meaning), and the origin of information.

in a fine-grained manner: It uses five different features for the `speciesURN` and species `name`, `compartmentURN` and compartment `name`, `reactionURN` and reaction `name`, `parameterURN` and parameter `name`, as well as `eventURN` and event `name` to support very specific queries on model constituents. The index in addition considers the relation of the annotated constituent and the annotation itself, by relating the qualifier information to the component.

Features of the user generated content dimension

The features of the user generated content dimension and where they are obtained from are shown in Table 7.7. When using the ranked retrieval approach inside a particular software framework, the users may provide tags for the submitted models and model versions. The tags can be stored in line with the model files (as proposed for *mDB*). The keywords feature then indexes the tags. It is regarded a valuable feature, as it is manually provided information, usually by domain experts.

If representation formats furthermore provide a textual description of the model and its constituents, for example the SBML `<notes>` tags, it is also indexed for later use in the search process.

Features of the experiment dimension

The features of the experiment dimension and where they are obtained from are shown in Table 7.8. Values for the experiment-related features are available from both, *eDB* (Section 6.6) and *sDB* (Section 5.4.3). Experiment types are extractable from both databases (e. g. "uniform time course" in *sDB* and "Western blot" in *eDB*). The simulation approach (`kisao` feature) and `behavior` are both extractable for simulation experiments stored in *sDB*. The `kisao` feature allows to limit a search to particular simulation approaches such as "discrete approaches", or even to a set of simulation algorithms, such as "models simulateable with Gillespie approaches". The `behavior` feature furthermore allows to restrict the search to models showing a

feature name	**meaning**	**origin**
experimentID	experiment ID	experimentID (*sDB*) / experimentURL (*mDB*)
experimentType	type of experiment	experimentType (*sDB*)
kisao	simulation approach	kisaoID (*sDB*)
behavior	observed behavior in simulation	teddyID (*sDB*)

Table 7.8.: Features in the experiment dimension with the feature name, its interpretation (meaning), and the origin of information.

particular desired behavior during simulation such as "steady-state". The fact that an experiment exists for a model does in addition enhance the model's value, as it does mark the model as "simulateable" for at least one certain parametrization. Therefore, the similarity function boosts models for which links to simulation experiments exist. It is assumed that a number of 1-5 experiments for a given model positively effects its relevance. That is why, a similarity function giving a very low punishment to models with 1 or more experiments assigned (> 0.6) and a high punishment (0.2) to models with no experiment assigned is a first reasonable approach to extend the weight concepts.

7.3.3. Structure information

The above mentioned dimensions and associated features only allow for the use of information retrieval queries. They do not support *structured queries*. However, structural information encoded in bio-models is important to answer questions such as "Find me all models that encode the species `lacI` as a reactant inside the compartment `cell`", or "How many models with reactions of two reactants exist in the model?".

From the network structure (structure axis in the coordinate system in Figure 6.1 on page 125), information on "connected" species can be inferred. The information is extractable from the representation formats. For example, reactions in SBML have specified `reactants` and `products` each linking to a species definition. That information, if kept in the database relations, can be used to enable above mentioned queries. The database relation model in Figure 6.3 on page 138 shows a solution to the problem: A general component entity keeps all information on the model constituents. Each component has a particular `role` assigned. Via the `isLocatedIn` and `participatesIn` relations, the structure can be queried using SQL queries.

The relation model in Figure 6.3 on page 138 shows that also information on the number of compartments in a particular model can be calculated by determining all

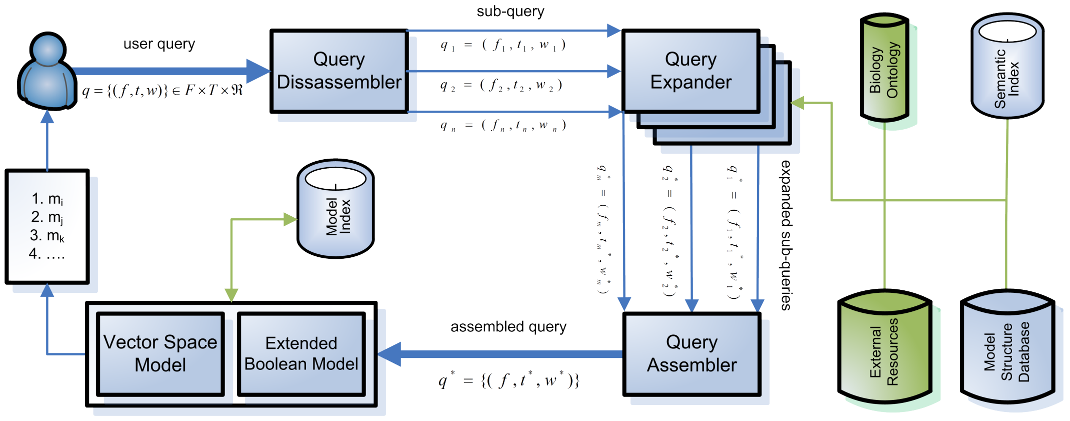


Figure 7.1.: The conceptual architecture of the proposed ranked retrieval for bio-models [Henkel et al. 2011].

components of a particular model that have the attribute `role` assigned with the value `compartment`.

7.3.4. Conceptual architecture

The developed approach to ranked retrieval uses an architecture that builds on different IR techniques (including text retrieval, stemming, term-frequencies) and types of meta-information (including ontologies, model meta-information, and model-related meta-information). The solution is based on the proposed measures for meta-data-based similarity, content-based similarity, and semantic-description-based similarity (as introduced in Section 2.5). MBSM techniques incorporate MIRIAM required meta-information about the model and its constituents in the ranking. CBSM techniques are applied by incorporating a model's low level features such as encoded species. SDSM techniques are applicable when evaluating user-provided tags and keywords describing the model and its related information.

Figure 7.1 shows the conceptual architecture. Its main building blocks are the query disassembler, the query expander, the query assembler, and the ranking and retrieval system. A first version of that concept had been proposed by Henkel [2009], and has since then been extended. Detailed descriptions of the architecture can be found in [Henkel et al. 2010, 2011]. The main ideas of the architecture are described in the following.

A user-given query q consists of a set of terms t that each have a feature f from the set of defined features F and a user-given weight w assigned, forming triples of (f, t, w). The concept supports two different kinds of queries, *query by value* and *query by model example* [Henkel et al. 2010].

Query by value (QBV) The user query q contains features f_i and free-text terms ρ. The user-given features in the query are a subset of all available features F.

Query by model example (QBME) The user query q contains a model that forms the basis for a search. The complete set of features $f_i \in F$ is aligned.

Query disassembler

The query q is first disassembled into sub-queries $q_1..q_n$. Each sub-query q_i represents a set of processed and normalized words (i. e. terms) t_i assigned to a particular feature f_i and provided with a user-given weight w, forming triples of a feature, associated terms, and weights $(f_i, t_1..t_n, w_1..w_n)$. Depending on the feature assigned to the sub-query, different data resources are queried to enrich the user-given query.

Query Expander

Sub-queries addressing the structure of a model are executed on the *Model Structure Database*. The matching model IDs are returned to the *Query Assembler* to enrich the original query. Sub-queries addressing model constituents are executed on the *Semantic Index*. External resources are queried to enrich the original query by related terms. The result is returned to the *Query Assembler*; the original user-given weights are updated based on the relevance of the incorporated ontological term. Sub-queries that do not need to be enriched are directly passed on to the *Query Assembler*.

Model Structure Database If information on the model's structure is available from the model repository, that information can be used to filter or expand the list of relevant models. Examples for model structure queries include "models with more than 3 compartments", or "models with the species Beta-Catenin as reaction product". All information can be gained from the reference database by executing SQL queries. The result is a set of model IDs. Those IDs restrict or extend the ranked result set in the retrieval and ranking system.

For example, to return all models that have the species Beta-Catenin as reaction product, all models encoding species that are known as representing `Beta-Catenin` and have the `role` product assigned, will be be added to the result set.

Semantic Index Queries regarding the existence of a particular model constituent in a model are assigned to the *Semantic Index*. As one prerequisite of this work, it is assumed that model constituents are described by annotation information m_A encoded in standard URIs. The search for a particular constituent will, however, typically be formulated using a string chain. The Semantic Index provides a mapping of all URIs used in the system on the string chains describing the URIs in the external resource. It furthermore relates the occurring URIs to a particular qualifier, which enables more specific queries (i. e. limited to a particular qualifier). Table 7.9 shows an extract of the Semantic Index as implemented by Henkel et al. [2010] in BioModels Database.

External Resources It is sometimes useful to include models containing *similar*, and not only identical, constituents in the search result (e. g. if a search resulted in only a few models) [Henkel et al. 2010]. Similar biological concepts can be identified with the *Biology Ontology*. Here terms of different existing ontologies are mapped on a common biology ontology [Schulz et al. 2010]; similar terms expand the query in the enrichment process. For example, a user searching for models encoding the species *caffeine* might also be interested in models encoding the species *xanthine* which is structurally related to caffeine.

Further information from external resources include information about associated simulation experiments and corresponding results (Section 5.4.3), as well as information on the experimental setups (Section 6.6). For example, the existence of a simulation or biology experiment for a model will enhance its confidence. The observed behavior during simulation is another example for information relevant for the ranking.

Query Assembler

The original query terms t_i and the expanded query terms resulting from the enrichment process q_{iexp} are then re-assembled into a single query $q*$, which is sent to the ranking and retrieval system.

Ranking and retrieval system

The ranking and retrieval system works in two different steps: First, the Extended Boolean Model retrieves all relevant models for the expanded query. Afterward, the Vector Space Model ranks the result sets (Section 2.5). The relevance ranking is additionally influenced by different types of *weights*.

ontology weights Terms that have been included in the assembled query $q*$ due to the evaluation of the Biology Ontology (i. e. terms that are related to the original search terms, but not identical) are weighted lower in the result ranking. The decrease of weight value depends on the distance between both terms in the ontology, and the term's depth in the tree.

URI	**qualifier**			**content**
	bqbiol_is	bqbiol_isVersionOf	...	
urn:miriam:obo.chebi. CHEBI:27732	BIOMD0000000241	BIOMD0000000241		caffeine chebi 27732 chebi home advanced browse ontology periodic ... moleculeschebimain caffeine chebi 116485 central nervous system stimulant caffeine receptor modulator 1,3,7-trimethyl-3,7 dihydro-1h-purine-2,6 1,3,7-trimethylxanthine msdchem d00528 kegg [..]
urn:miriam:kegg. compound:C07481	BIOMD0000000241			kegg compound c07481 c07481 compound caffeine 1,3,7-trimethylxanthine formula c8h10n4o2 mass structure remark d00528 source coffea arabica tax xanthines reaction r07920 27732 knapsack c00001492 [..]
urn:miriam:kegg. compound:C00048		BIOMD0000000221 BIOMD0000000222 BIOMD0000000219 BIOMD0000000218		kegg compound c00048 c00048 glyoxylate acid formula c2h2o3 mass structure reaction r00013 purine metabolism path caffeine glycine serine null_1 threonine metabolism [..]
...				...

Table 7.9.: Extract of the Semantic Index as implemented in [Henkel et al. 2010]: The URI column contains all URIs occurring in all models stored in BioModels Database. The content column contains the different string chains occurring in the external resource describing each URI. The qualifier columns link the URIs to the particular model ID they occur in, storing also how the URI relates to the model ID.

feature weights Each feature has a pre-defined weight assigned. For example, a search in the `speciesName` feature is less important than in the `speciesURN` feature. When concatenating the different similarity value for single features, results of important features are boosted higher than results of features with a lower importance.

user-defined weights Users might assign weights to each of their query terms. The weights can increase the importance of particular terms and are incorporated in the ranking.

common weight strategies The *Model Index* itself is used to incorporate weights derived from IR techniques, such as term frequency–inverse document frequency.

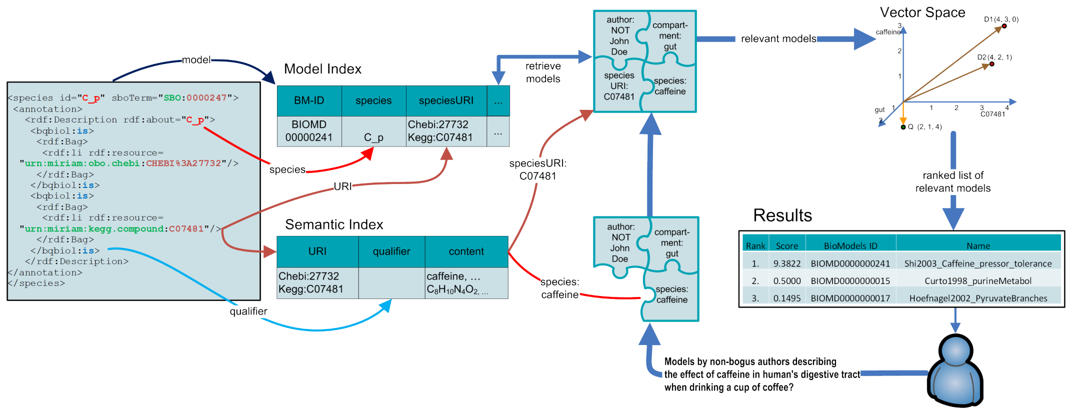
All weights assigned to the terms, which describe a model, are used to determine the model's position in the vector space. The ranking is then calculated based on the model vector's similarity with the query vector.

Model Index The *Model Index* contains references to all models $m_i \in M$. For each model, it stores information about the occurring model constituents, as well as meta-information. As such, the index contains a column for each identified feature (Section 7.3.2). The Model Index enables to restrict the search to a particular feature (e. g. "models with *Beta-Catenin* being an encoded *species*", or "models with *Frizzled* being the `author name`"). In the second example, a query that does not restrict the search term "Frizzled" to the author name will certainly return a major number of WNT signaling pathway models, as *Frizzled* is the name of a receptor in that pathway. However, to say that we were particularly interested in models by the *author* named Frizzled, we can restrict our query to the `author` feature in the index.

7.3.5. Sample query

While the introduced dimensions and features are conceptual, the introduced architecture has been implemented with a subset of the proposed dimensions, using a subset of the features. Figure 7.2 shows an example for the ranked retrieval of a set of models for a given user query, using the prototype implementation available from BioModels Database [Henkel et al. 2010]. User-given weights are not shown in the example for better readability.

Imagine a user was looking for "Models by non-bogus authors describing the effect of caffeine in the human digestive tract when drinking a cup of coffee". Using the



```
<species id="C_p" sboTerm="SBO:0000247">
 <annotation>
   <rdf:Description rdf:about="C_p">
    <bqbiol:is>
     <rdf:Bag>
      <rdf:li rdf:resource=
  "urn:miriam:obo.chebi:CHEBI%3A27732"/>
     </rdf:Bag>
    </bqbiol:is>
    <bqbiol:is>
     <rdf:Bag>
      <rdf:li rdf:resource=
  "urn:miriam:kegg.compound:C07481"/>
     </rdf:Bag>
    </bqbiol:is>
   </rdf:Description>
</annotation>
</species>
```

Model Index

BM-ID	species	speciesURI	...
BIOMD 00000241	C_p	Chebi:27732 Kegg:C07481	...

Semantic Index

URI	qualifier	content
Chebi:27732 Kegg:C07481		caffeine, ... $C_8H_{10}N_4O_2$, ...

Results

Rank	Score	BioModels ID	Name
1.	9.3822	BIOMD0000000241	Shi2003_Caffeine_pressor_tolerance
2.	0.5000	BIOMD0000000015	Curto1998_purineMetabol
3.	0.1495	BIOMD0000000017	Hoefnagel2002_PyruvateBranches

Models by non-bogus authors describing the effect of caffeine in human's digestive tract when drinking a cup of coffee?

Figure 7.2.: A sample scenario for a search using the ranked retrieval architecture proposed in Figure 7.1 (published in [Waltemath et al. 2011c]). The left hand side shows the index creation process, the right hand side exemplifies a query with its result ranking.

above architecture, the query can be formulated as *-author:(John Doe) +compartment:(gut) +species:(caffeine)* [Henkel et al. 2010]. The Query Disassembler creates three sub-queries:

- `-author:(John Doe)`
- `+compartment:(gut)`
- `+species:(caffeine)`

In the Lucene syntax used here, the `-` indicates that a term *must not* occur, a `+` indicates that the term *must* occur, otherwise (no leading sign) a term *should* occur [Hatcher and Gospodnetic 2004, p. 93]. The Query Expander expands the sub-queries, if possible: The `author` feature is not expanded. The second sub-query `+compartment:(gut)`, however, can be expanded using the Semantic Index. The string "gut" is looked up in the `content` column of the Semantic Index. If the term is found, then all URIs containing it are added to the query. In this example, "gut" is not found in any of the `content` columns, and therefore the sub-query cannot be extended. The third sub-query `+species:(caffeine)` can also be expanded. The string "caffeine" is looked up in the `content` column of the Semantic Index. For this term, two occurrences are found: The URN corresponding to "caffeine", and the URN corresponding to the chemical formula "$C_8H_{10}N_4O_2$", which is the chemical notation of caffeine. Both URNs, `Chebi:27732` and `Kegg:C07481` (abbreviated in the Figure for convenience), are candidates for a query expansion. However, due to the additional restriction that the qualified relation must be an `is` relation, only the Kegg URN forms a new sub-query. `Chebi:27732` occurs as an annotation using the `isVersionOf` qualifier. Consequently, the sub-query *+(species:(caffeine) speciesURI:C07481)* is sent to the Query Assembler.

The subqueries are re-assembled into the expanded query q^* and then sent to the retrieval and ranking system. The Extended Boolean Model is used to determine the relevant models fulfilling all the conditions. The result is a set of internal ModelIDs (Index) and associated feature values. The relevant models are positioned in the vector space ($D1, D2$ in Figure 7.2) and a relevance ranking is determined. A ranked list of models is returned to the user. The similarity value indicates how similar the found model is to the query.

7.4. Test framework

The working of ranked retrieval on computational biological models has been shown in [Henkel et al. 2010], using the example of SBML models stored in BioModels Database, and the ranking function implemented by Henkel [2009]. Already during the conceptual development of the ranked search system, became obvious that the fine-tuning of the ranking function and it's parameters demands a lot of effort. Different settings for a ranking functions, including initial features weights or boosts, must be tested on the data set to gain a satisfying ranking result. Furthermore, different ranking functions might be applicable to a data set and therefore their performances need to be compared. More than that, the CSB field uses a number of different representation formats for different kinds of models (Section 2.2). Therefore, a ranking function needs to be tested on a number of data sets, as it might perform differently well depending on the nature of the encoded data [Waltemath et al. 2011c].

This thesis proposes a framework for testing different similarity measures on different sets of computational biological models, called the *Searching fOr Models annotated with Biological Information* (*Sombi*) framework. The basic components of the framework architecture are shown in Figure 7.3. Each rectangle in the Figure corresponds to a so-called *module class* representing a particular software class. For each such module class, a number of different *modules* may exist which can be added interchangeably to the framework. The modules are interconnected via well-defined interfaces, allowing for data exchange among them. The initial version of the *Sombi* test framework has been implemented during the course of a software project at Rostock university. The code is available on `http://sombi.sourceforge.net/`. The framework prototype is written in Java and contains interfaces for the different *Sombi* modules, providing general module classes that can be refined for the different written modules. The framework contains the following module classes, including

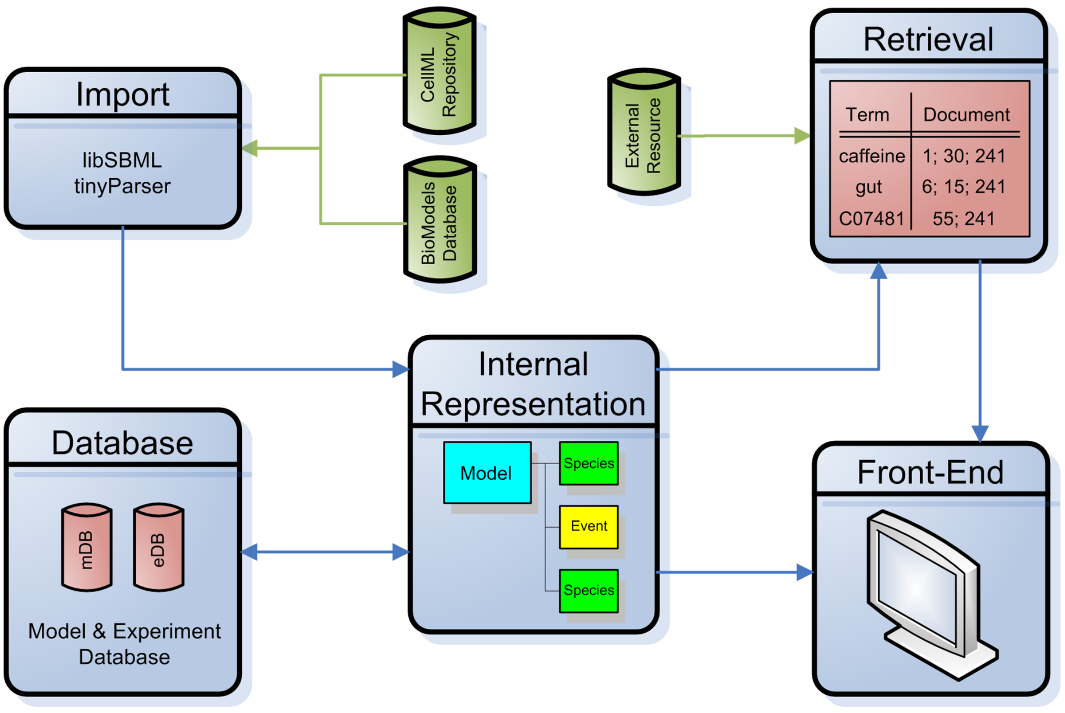


Figure 7.3.: The *Sombi* framework architecture [Waltemath et al. 2011c]

the mentioned modules.

Internal Representation module class The central module class is the *Internal Representation module class*. It provides the model in an internal Java object and XML representation. The internal representation format has been based on ideas of the model parser for the *mDB* (Section 6.2.3). The XML Schema definition of the internal representation shown in Appendix A.5.1 has first been developed by Hälke [2009] and then been extended. The internal representation is the basis for all other modules. It ensures that all information necessary for storage, retrieval and ranking functionality, and for the front-end is provided. A model's internal representation is created during the model import process.

Import module class The *Import module class* provides modules for the import of model from a particular model database or model repository for use in the *Sombi* framework. While the framework contains a module for SBML model import, it is generally expected that representation formats and model resources implement their own parser for the data mapping onto the internal representation format. The SBML parser available from the *Sombi* framework is called *tinyParser* [Hälke 2009]

(Section 6.2.3). It parses and imports SBML models in all levels and versions. Parsers for other model formats, including πML and CellML can be developed by extending the tinyParser. It is also possible to reuse existing parsers, such as `libSBML` and adapt their output.

Database module class The *Database module class* stores model information. It imports the information stored in the *internal representation*, but also stores the original model files, and system data. *Sombi*'s standard database is the *mDB* (Section 6). *mDB* contains imported SBML models from BioModels Database. To realize future incorporation of experimental and simulation result data, interfaces for *eDB* (Section 6.6) and *sDB* (Section 5.4.3) are planned.

Retrieval module class The *retrieval module class* provides ranking functions for the retrieval of models in the database module. The use of more than one ranking function enables the comparison of different ranking strategies. The provided ranking module incorporates information provided by *Sombi*'s internal representation format. *Sombi* comes with two different ranking functions: The first one is a Lucene implementation, combining the Standard Boolean Model and the Vector Space Model, similar to the approach introduced in Section 7.5. A weight matrix allows to adjust the parameters of the ranking function for testing. The second function is based on a simple XML parser which extracts the XML element names and indexes them. However, users can add own implementations of ranking functions to the system.

Front-End module class The front-end module presents the ranking results to the user. It contains an explanation of how the similarity value was calculated. A simple output has already been provided by Henkel [2009]. The front-end furthermore should present the differences between the query and the retrieved models, as well as the differences among the retrieved models. In the current implementation, the in- and outputs of the different modules are command-line based; a portlet-based front-end module is under development. Due to the modular design, plugging in additional front-end implementations is easily possible.

7.5. Implementation

Ranked retrieval of bio-models has so far been applied to the BioModels Database search system and on the *Sombi* framework.

Ranked retrieval in BioModels Database A ranked retrieval approach in a model repository such as BioModels Database enhances the usability of the system and results in a better reuse of existing information [Henkel et al. 2010]. The ranking concept introduced in this chapter has been implemented in BioModels Database by Ron Henkel [Henkel et al. 2010]. The advantage of using BioModels Database lies in the amount of stored models – currently 630 models[1] are available in SBML encoding. Models from the curated branch are annotated. Consequently, they provide sufficient meta-information for a thorough testing of the ranking and retrieval system. Analyzing the stored information together with the *BioModels.net team* led to tentative weights for the different features (Table 7.10) and helped on pinpointing the importance of different qualifiers [Henkel et al. 2010].

The implementation extends the BioModels Database standard search by including features in the search process, by weighting different information, and by ranking the results according to the user query. Support for both, QBV and QBME has been provided. QBV allows to either perform a free text search that incorporates all features, or alternatively a more sophisticated search selecting features of the different dimensions to be searched. For instance, a user might search for models by a certain author, or for models encoding a particular "species". The different sub-queries of a query might be weighted according to the implemented standard feature matrix. Especially queries against the constituent dimension can be enriched or limited.

The index incorporates the available qualifiers (Appendix A.1). Depending on the qualifier linked to a URI that URI might be weighted higher (e. g. `isA`) or lower (e. g. `hasPart`). Additionally, the user can specify the importance of each search term to influence the result 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 [Henkel et al. 2010]. QBME uses a model as the input to the search system. All values of the extracted features are queried against the repository. A ranked list of best matching models is retrieved. Enriched queries are switched off, as the example model itself provides sufficient contextual information.

The search system can be tested via the BioModels Database demo system on `http://www.ebi.ac.uk/biomodels-demo/`. The ranking incorporates the sub-set of earlier identified features that can be extracted from the current BioModels Database design (Table 7.10). Some features, however, could not be used as the current database schema does not envisage their storage (e. g. simulation experiment information or structure information).

To date, the model index incorporates 454 models with 140977 terms separated

[1]as of 30 September 2010

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		
	publicationText	2.5	data	ID	1
				additionalID	1
				path	1
				content	1

Table 7.10.: Features incorporated in the standard ranking function. Sorted by dimension and with assigned standard weight (w). Adapted from [Henkel et al. 2010]. `w` denotes the standard weight of the feature in the implemented ranking function.

into 25 different features. The semantic index contains 2261 URIs with 409124 terms. The biology ontologies used for query expansion are *NCBI Taxonomy*, *GO*, *ChEBI*, *KEGG Compound* and *KEGG Reaction*[2]. The Lucene Framework is integrated in the search system to create, maintain and search both, the Model Index and the Semantic Index. It provides retrieval functionality based on the Extended Boolean Model; the ranking is based on the Vector Space Model. To implement the retrieval and ranking process described above, Lucene has been extended to support the necessary indices and sources.

A detailed example for the execution of a query has been given in [Henkel et al. 2010] (Appendix A.5.3).

Sombi Framework The *Sombi* concept has been prototypically realised as a modular framework. All code is available from the sourceforge site[3]. The current implementation consists of the following modules:

Import module class The Import module class contains a module for the parsing of SBML models with the *tinyParser*. The parser extracts all necessary information from the SBML model and transforms that information into the internal representation format.

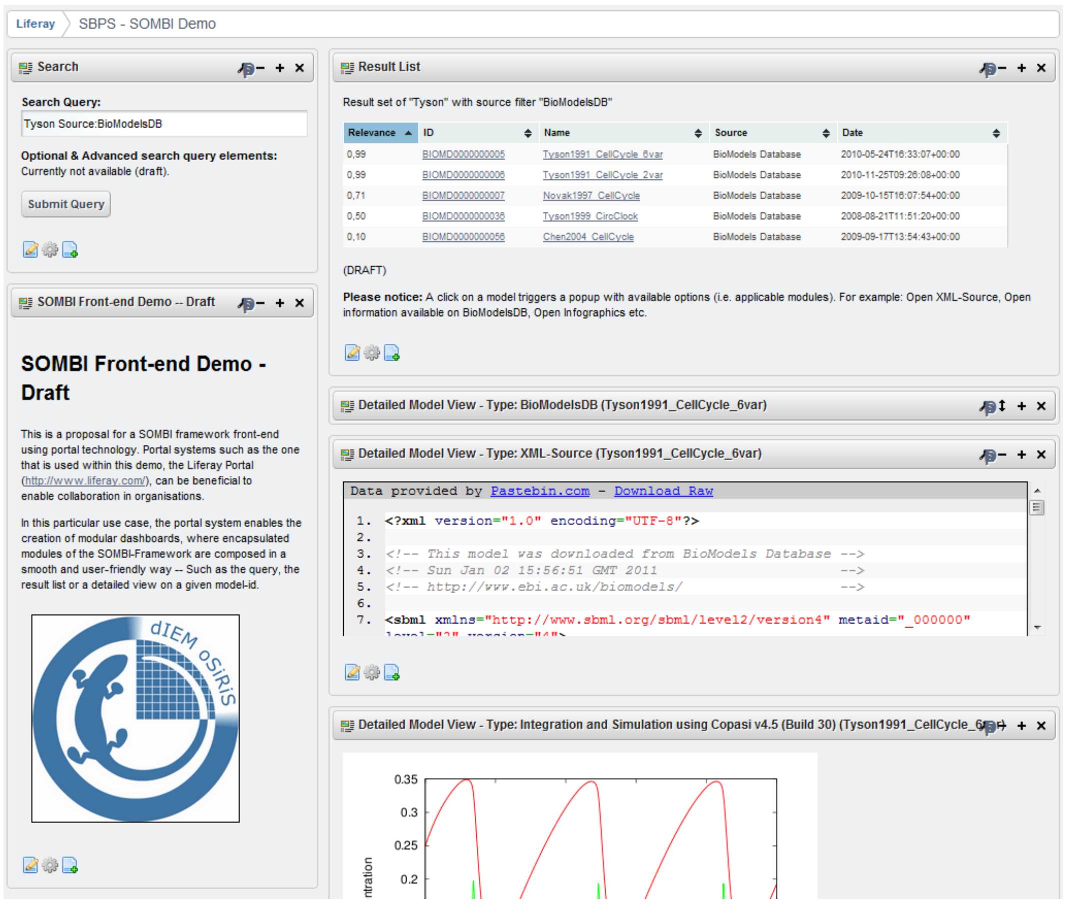
Database module class The Database module class provides a MySQL implementation of the *mDB* relation model, including the DDL and the commands to fill the database instance with standard values for formalisms and qualifiers. The module is capable of reading the IRF and importing its information into the database.

Retrieval module class The Retrieval module class has three different modules for the ranking which are all based on the IRF. The first module uses a full text index that considers the IRF as a text document and extracts from it attribute and element values. The second module uses the features also implemented in BioModels Database and applies the combination of Boolean model/Vector space model. The third module uses the same indexing method as the one implemented in BioModels Database, but distinguishes only six different model components.

The framework itself coordinates the integration of the different modules from the existing classes. A predefined XML-format for the framework configuration is provided together with some standard examples.

[2]as of 14 April 2010

[3]`http://sourceforge.net/projects/sombi/`, last accessed 20 March 2011.



Optional & Advanced search query elements:
Currently not available (draft).

SOMBI Front-end Demo - Draft

This is a proposal for a SOMBI framework front-end using portal technology. Portal systems such as the one that is used within this demo, the Liferay Portal (http://www.liferay.com/), can be beneficial to enable collaboration in organisations.

In this particular use case, the portal system enables the creation of modular dashboards, where encapsulated modules of the SOMBI-Framework are composed in a smooth and user-friendly way -- Such as the query, the result list or a detailed view on a given model-id.

Result set of "Tyson" with source filter "BioModelsDB"

Relevance	ID	Name	Source	Date
0,99	BIOMD0000000005	Tyson1991_CellCycle_6var	BioModels Database	2010-05-24T16:33:07+00:00
0,99	BIOMD0000000006	Tyson1991_CellCycle_2var	BioModels Database	2010-11-25T09:26:08+00:00
0,71	BIOMD0000000007	Novak1997_CellCycle	BioModels Database	2009-10-15T16:07:54+00:00
0,50	BIOMD0000000036	Tyson1999_CircClock	BioModels Database	2008-08-21T11:51:20+00:00
0,10	BIOMD0000000056	Chen2004_CellCycle	BioModels Database	2009-09-17T13:54:43+00:00

(DRAFT)

Please notice: A click on a model triggers a popup with available options (i.e. applicable modules). For example: Open XML-Source, Open information available on BioModelsDB, Open Infographics etc.

Detailed Model View - Type: BioModelsDB (Tyson1991_CellCycle_6var)

Detailed Model View - Type: XML-Source (Tyson1991_CellCycle_6var)

```
Data provided by Pastebin.com - Download Raw

<?xml version="1.0" encoding="UTF-8"?>

<!-- This model was downloaded from BioModels Database -->
<!-- Sun Jan 02 15:56:51 GMT 2011                      -->
<!-- http://www.ebi.ac.uk/biomodels/                   -->

<sbml xmlns="http://www.sbml.org/sbml/level2/version4" metaid="_000000"
```

Detailed Model View - Type: Integration and Simulation using Copasi v4.5 (Build 30) (Tyson1991_CellCycle_6var)

Figure 7.4.: Preview of the *Sombi* Front-end standard module [Waltemath et al. 2011c].

Modules for the Front-end module class are under development. One proposal for a front-end architecture based on *portlets* is under development [Weitzel 2011 (to appear]. The front-end integrates different views on model management tasks, including the model version history, the model meta-information display, and simulation experiment descriptions linked with the model (Figure 7.4).

7.6. Summary

Results

This chapter applied standard IR techniques on bio-model retrieval. The work assumed the existence of encoded meta-information in the bio-models. Here the meta-information was obtained from *mDB* (Section 6.3.3).

I argue that an important step towards better model reuse is the ability to retrieve relevant models for a query. The incorporation of meta-information on top of the model's structure is a key issue [Köhn et al. 2009]. Meta-information includes biological details about the modeled system, but also applied simulation experiments, model parametrizations, and details about the computational model such as available model versions, the model author or reference publications. Another important aspect is the consideration of structural information.

The application of IR techniques on bio-model retrieval improves existing search facilities and results in a fine-grained similarity search. To show the feasibility of the ranked retrieval system, parts of the concept have been implemented as a new demo search system of BioModels Database.

Discussion

Evaluation So far, no evaluations have been done to measure the efficiency of the retrieval approach. One major problem is the fact that the implementation provided in [Henkel et al. 2010] was the first application of ranked retrieval to the bio-models domain. Consequently, no systems existed for a comparison. The only reference that has been used for testing is the former BioModels Database search system, which is still available from the web site[4]. Sample searches with members of the BioModels Database curator team have shown that the retrieval approach returns more relevant models for a given query. For example, a search for models dealing with `caffeine` in the original BioModels Database search system returns one result model (i. e. `BIOMD0000000241`). The new search system, however, returns 15 models of which three models are relevant. Indeed, the first model is `BIOMD0000000241` (similarity value of 0.3914), the model on caffeine pressor tolerance, the second model is `BIOMD0000000015` (0.0405), a model involving purine which is chemically related to caffeine. The third model, `BIOMD0000000060` (0.0334), models the ryanodine receptor which is also known as a caffeine-sensitive calcium-release receptor. While those examples show the use of our search approach, they cannot be measured in terms of a Gold standard. The development of such a standard is underway though. Current works in the BioModels Database team include the tagging of models in the system. This will enable the definition of sample queries and expected results for those queries in order to form a test set which can then be used to evaluate different approaches to model retrieval.

The introduced *Sombi* framework serves as a framework for the testing of different ranking functions on different model resources. The data resource used in *Sombi* is *mDB* which contains all models from BioModels Database and stores them in

[4]`http://www.ebi.ac.uk/biomodels-main/search`, last accessed 22 November 2011.

the meta-information focused manner. Three different ranking functions are available. As the framework is in prototype stage, it lacks evaluations of the different incorporated ranking functions and model sources though.

Usability *mDB* to date does not sufficiently inform the user about why a model was found. The current interface is prototypical and as such does not comply with current usability standards. Technically, all needed information is available from the retrieval system, but it is not presented to the user. For example, the search for models by the author *Elowitz* that involve the *lactose inducer (LacI)* gives a great number of results for which it would be useful to have a way of communicating:

- Which models are exact matches (models by Elowitz that really encode LacI)
- Which models are models that encode LacI, but are not designed by Elowitz
- Which models are Elowitz models, but might not include LacI

Flexible weights First observations have shown that the ranked retrieval approach is feasible, but needs to be extended towards a more flexible concept for weights and user preferences, as the different user groups working with the data have varying demands. As mentioned in Section 7.3.2, the identified features are of different importance for the ranked retrieval, depending on the kind of user. For example, a user intending to simulate a model in a given simulation environment may assign a particular importance to the `algorithm` feature. A systems biologist searching for a model of a particular biological system, however, may decide to disregard the model's encoding format, but instead assign a particular importance to the biological background the model is able to provide; he might be flexible on the simulation environment to use.

Demands regarding model retrieval differ for user classes. One way of solving is to allow users to boost particular query terms. However, a second idea is to encode standard feature matrices for the different user groups (i. e. user profiles). For the moment, feature weights are preset. They might be adjusted using relrevance feedback methods in the future.

Incorporated similarity features The concept incorporates meta-information on the model, but as well on existing model versions, on existing simulation experiments, on the observed model behavior and so on. Parts of the information can be encoded in standard formats (Section 3.1.1).So far, only parts of the concept for integrating meta-information and model-related information have been incorporated in the implementations, and evaluations are necessary to put the value of the approach in numbers. Further investigations in standardization formats and ontologies are necessary.

Two-step retrieval Besides the approach to encode and use the semantics of biomodels, there have also been attempts to use the model's structure to facilitate model reuse [Kell and Mendes 2008], for example as components of coupled systems [Röhl and Uhrmacher 2005]. However, evaluating the structural information of the XML file alone is far too general to define a similarity mapping between different models. One solution for models stored in *mDB* is a two-step search process: First, relevant models are identified based on their matching annotations. Secondly, relevant models are analyzed on the structure level to find overlapping or similar model constituents. This work only realizes the first step. The second step needs to be implemented in future versions, probably requiring to move to a storage approach that is tailored towards XML storage and fulfills the specific requirements described in Section 2.4.

8. Conclusions

If we knew what it was we were doing, it would not be called research, would it?

(Albert Einstein)

Computational support has become an integral part of studying biological phenomena. The number and variety of computational models in public databases are increasing rapidly, demanding new methods for efficient model search.

This thesis investigates methods and techniques for enhanced *model retrieval*. The consequence will be augmented reuse of existing models. Figure 8.1 shows the particular contributions of this thesis to the research area, as well as their interconnections. Major aspects are (1) the enhancement of meta-information encoding (Chapter 4), (2) the development of a format for the exchange of simulation experiments (Chapter 5), (3) the development of a format-independent database for model storage (Chapter 6), and (4) the application of ranked retrieval on bio-models (Chapter 7). The main XML model encoding format used for this work is SBML as a wide range of SBML models is available from public resources for testing.

I improved the encoding format of model annotations in SBML. In this work I proposed an improved version of the SBML annotation scheme, referred to as the *annot package*. Resulting from first experiences with the retrieval implementation and from an evaluation of the current SBML Level 3 Core annotation standard, we proposed an extended annotation concept together with colleagues at different institutes (e. g. Humboldt University Berlin, Manchester University, Newcastle University, European Bioinformatics Institute). The major outcome of the project is an extension to the current annotation scheme, instantiated in a stand-alone SBML "package". The Annotation package takes advantage of the expressiveness provided by RDF. Allowing full RDF in SBML annotations now enables, for example, distinguished relations between annotations and the annotated SBML element, annotations of XML attributes and statements about statements. Consequently, biological knowledge can be more precisely encoded in future SBML models. The use of RDF statements also allows for evaluation by existing reasoning tools.

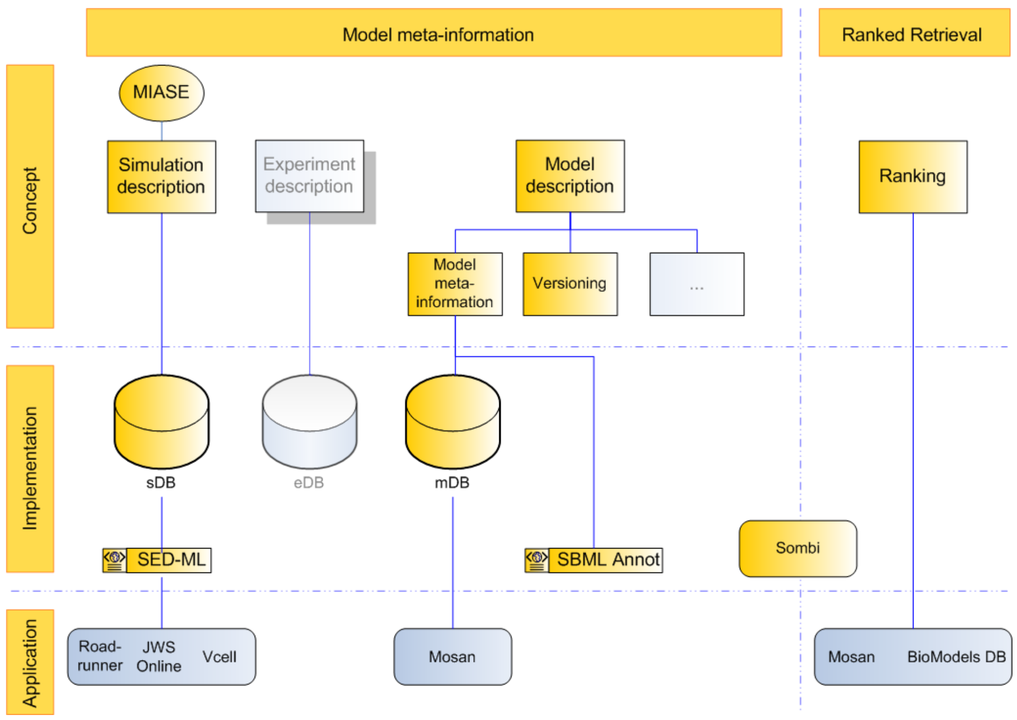


Figure 8.1.: Investigated aspects, realizations and applications of model storage and retrieval.

I developed standards for simulation experiment encoding. Standardization efforts for the encoding of reusable information are important. They are integral to some areas of computational systems biology (e. g. kinetic modeling), but not yet established in all of them. Particularly standard formats for the encoding of simulation experiment descriptions are missing. I developed an ontology, called *KiSAO*, for the characterization and classification of existing simulation algorithms used for the simulation of kinetic models. I also developed a format for the encoding of simulation experiment descriptions, called *SED-ML*. It is based on a set of Minimum Information guidelines for simulation experiments, called *MIASE*. I have shown that simulation experiments encoded in SED-ML can be reproduced, even in different simulation environments. The proposed language *SED-ML Level 1 Version 1* is considered a standard format for simulation experiment encoding not only by the SBML community, but also by the CellML community and the NeuroML community.

I developed a format-independent, meta-information based storage solution for bio-models. While SBML is the *de-facto* standard for the encoding of biological models, other formats co-exist. Model repositories restrict themselves to particular representation formats: SBML-encoded models are typically stored in BioModels Database, but also in the JWS Online model repository. CellML-encoded models are stored in the CellML Model repository. Models with a strong focus on neuronal systems might be found in ModelDB, and so on. However, a model's actual XML encoding is often marginal. On the contrary, a comprehensive collection of existing knowledge about the model (i. e. meta-information) is of interest.

I have therefore designed a *format-independent model database*, called *mDB*. It realizes a meta-information-centric storage approach in a relational database schema. The result is a system that allows for formalism-independent storage and retrieval of computational biology models. It also incorporates model meta-information and model-related information such as associated simulation experiments. The fine-grained meta-information storage comprises a thorough versioning system to follow the evolution of stored models and enables model retrieval, model visualization, model versioning, and similar tasks.

I designed a ranked retrieval approach to model search. Current model search systems rely on data retrieval techniques. They only partially incorporate meta-information and do not rank their search results.

As one part of my thesis I present the application of Information Retrieval methods on *bio-model retrieval*. I developed a concept for ranked model search on the basis of identified model *features*. The features were derived from the data, meta-data, information, and meta-information stored in *mDB*. The retrieval system supports query term boosts, feature weights and query expansion with knowledge from external resources.

The meta-information-based search allows for a greater range of more specific queries. It also returns more potentially relevant models for a query, including similar, and not only identically matching results. The conceptual design benefits from the incorporation of model-related simulation experiments and results, as well as structural information extractable from the XML tree structure that maps the biological network structure. A first implementation has been realized as part of a Diploma thesis that I supervised. A second implementation has been integrated in the BioModels Database search system.

Figure 8.1 shows how the different contributions are aligned. *mDB* is the central database for testing the storage and retrieval concepts. The *ranking* approach has first been tested on that database, before being implemented as a test search in *BioModels Database* . A prerequisite for testing different ranking functions is *Sombi* – a modular framework that can plug in different model repositories and ranking functions, and allows to compare the outcomes of their combinations. Evaluations of the *mDB* and ranking approaches led to the effort of defining an extended annotation scheme for SBML models (*SBML annot*). The visualization tool *Mosan* uses the *mDB* search to import and display ranked results; the advantage being that Mosan is capable of handling multiple models and their experiments. *mDB* is connected to an experimental database *eDB* that I developed to support the storage of biology experiments within the research training school. In the future, the experiments shall be annotated by domain experts, and then be incorporated in the search and ranking process. Once that is done, the experiments in *eDB* will also be retrieved for models displayed in Mosan. The third database designed in line of this thesis is the simulation experiment database *sDB* that links experiments to models in *mDB*. It is capable of storing SED-ML simulation experiment descriptions. The working of SED-ML has been shown in the simulation environments *Roadrunner*, *JWS Online simulator* and *VCell* by the software developers.

I am convinced that the promotion of standardization efforts is crucial to systems biology projects as standards enable the exchange and reuse of information. Biological systems are too complex to be studied by isolated groups, and large scale collaborations are a prerequisite to successful research. I would like to cite a colleague here who said that "the synthesis of information has generated new insights, and therein lies the power – and the beauty – of Systems Biology"[1]. Standardization is the very basis to help researchers synthesize their work and knowledge. Standardization thereby generates new science.

[1]M. E. Stefan, "On the function of calcium-regulated allosteric devices in synaptic plasticity", dissertation (2009)

A. Appendices

A.1. Qualifiers

A.1.1. Biomodels.net qualifiers

The following are the qualifiers proposed by the *biomodels.net* effort (taken from the biomodels.net web site[1]). They are distinguished in model qualifiers and biology qualifiers.

Model qualifiers

These kind of qualifiers define the relationship between a modelling object and its annotation. The used namespace prefix is *bqmodel*:

qualifier	**description**
is	The modelling object represented by the model component is the subject of the referenced resource. For instance, this qualifier might be used to link the encoded model to a database of models.
isDerivedFrom	The modelling object represented by the component of the encoded model is derived from the modelling object represented by the referenced resource. For instance, they can be the fruit of a refinement or their adaptation for usage in a different context.
isDescribedBy	The modelling object represented by the component of the encoded model is described by the referenced resource. This relation might be used to link a model or a kinetic law to the literature that describes this model or this kinetic law.

Table A.1.: biomodels.net model qualifiers

Biology qualifiers

These kind of qualifiers define the relationship between a biological object represented by a model element and its annotation. The used namespace prefix is *bqbiol.*

qualifier	**description**
encodes	The biological entity represented by the model component encodes, directly or by transitivity the subject of the referenced resource.
hasPart	The biological entity represented by the model component includes the subject of the referenced resource, either physically or logically. This relation might be used to link a complex to the description of its components.
hasProperty	The subject of the referenced resource is a property of the biological entity represented by the model component. This relation might be used when a biological entity has a given activity or exerts a specific function.
hasVersion	The subject of the referenced resource is a version or an instance of the biological entity represented by the model component.

[1]Available from `http://www.biomodels.net/qualifiers/`, last accessed 1 February 2011.

is	The biological entity represented by the model component is the subject of the referenced resource. This relation might be used to link a reaction to its exact counterpart in KEGG or Reactome for instance.
isDescribedBy	The biological entity represented by the model component is described by the referenced resource. This relation should be used for instance to link a species or a parameter to the literature that describes the concentration of the species or the value of the parameter.
isEncodedBy	The biological entity represented by the model component is encoded, directly, or by transitivity, by the subject of the referenced resource.
isHomologTo	The biological entity represented by the model component is homologous to the subject of the referenced resource, i. e. they share a common ancestor.
isPartOf	The biological entity represented by the model component is a physical or logical part of the subject of the referenced resource. This relation might be used to link a component to the description of the complex is belongs to.
isPropertyOf	The biological entity represented by the model component is a property of the referenced resource.
isVersionOf	The biological entity represented by the model component is a version or an instance of the subject of the referenced resource.
occursIn	The biological entity represented by the model component takes place in the subject of the reference resource.

Table A.2.: biomodels.net biology qualifiers

A.1.2. Proposed revised biomodels.net qualifiers

The following is a list of proposed revised and extended qualifiers for bio-model annotation. The list is published in [Waltemath et al. 2011d, p. 14]. The new qualifiers have been discussed on the *sbml-annot* mailing list; there is as of now no final agreement.

qualifier	**description**
bqmodel:is	bqmodel:identity
bqmodel:isDerivedFrom	bqmodel:progenitor, bqmodel:antecedent, bqmodel:ancestor, bqmodel:basis, bqmodel:base, bqmodel:foundation, bqmodel:origin
bqmodel:isDescribedBy	bqmodel:description
bqbiol:hasPart	bqbiol:part
bqbiol:hasProperty	bqbiol:property
bqbiol:hasVersion	bqbiol:version
bqbiol:is	bqbiol:identity
bqbiol:isDescribedBy	bqbiol:description
bqbiol:isHomologTo	bqbiol:homolog
bqbiol:isEncodedBy	bqbiol:encoder
bqbiol:encodes	bqbiol:encodement
bqbiol:isPartOf	bqbiol:encompassment, bqbiol:assembly, bqbiol:partship, bqbiol:parthood, bqbiol:whole, bqbiol:meronym
bqbiol:isPropertyOf	bqbiol:bearer, bqbiol:carrier
bqbiol:isVersionOf	bqbiol:consociate, bqbiol:cohort, bqbiol:superclass, bqbiol:hyponym
bqbiol:occursIn (physical containment)	bqbiol:encompassment, bqbiol:containment
bqbiol:occursIn (taxonomic instantiation)	bqbiol:instantiation

Table A.3.: Proposed revised biomodels.net qualifiers

A.2. Bio-Models

A.2.1. Leloup Goldbeter 1999 (original and modified SBML encoding)

The original SBML encoding of the model describing the publication “Chaos and birhythmicity in a model for circadian oscillations of the PER and TIM proteins in Drosophila“ [Leloup and Goldbeter 1999] is available from `urn:miriam:biomodels.db:BIOMD0000000021`. The modified encoding of the model is shown in Listing A.1. Modifications compared to the original file are[2]:

- Namespaces omitted
- Species `id="T2"` renamed to `name="A"`
- Reaction `id="T0_to_T1"` renamed to `name="r1"`
- Reaction `id="T1_to_T2"` annotations removed
- SboTerm `sboTerm="SBO:0000179"` added to reaction `id="T2_degradation"`

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" metaid="metaid_0000001"
    level="2" version="1">
 <model metaid="metaid_0000002" id="Leloup1999_CircClock_periodic"
  name="Leloup1999_CircClock">
  <listOfUnitDefinitions [..] />
  <listOfCompartments [..] />
  <listOfSpecies>
   <species metaid="metaid_0000013" />
   <species metaid="metaid_0000014" id="T0" name="TIM Protein (
       unphosphorylated)" compartment="Cell" initialConcentration="0">
    <annotation>
     <rdf:RDF>
      <rdf:Description rdf:about="#metaid_0000014">
       <bqbiol:isVersionOf>
        <rdf:Bag>
         <rdf:li rdf:resource="urn:miriam:uniprot:P49021" />
        </rdf:Bag>
       </bqbiol:isVersionOf>
      </rdf:Description>
     </rdf:RDF>
    </annotation>
   </species>
   <species metaid="metaid_0000015" />
   <species metaid="metaid_0000016" id="T1" name="TIM Protein (mono-
       phosphorylated)" compartment="Cell" initialConcentration="0">
    <annotation>
     <rdf:RDF>
      <rdf:Description rdf:about="#metaid_0000016">
       <bqbiol:isVersionOf>
        <rdf:Bag>
         <rdf:li rdf:resource="urn:miriam:uniprot:P49021"/>
        </rdf:Bag>
```

[2]model version 29 September 2010

```
     </bqbiol:isVersionOf>
    </rdf:Description>
   </rdf:RDF>
  </annotation>
 </species>
 <species metaid="metaid_0000017" [..] />
 <species metaid="metaid_0000018" id="T2" name="A" compartment="Cell"
     initialConcentration="0">
  <annotation>
   <rdf:RDF>
    <rdf:Description rdf:about="#metaid_0000018">
     <bqbiol:isVersionOf>
      <rdf:Bag>
       <rdf:li rdf:resource="urn:miriam:uniprot:P49021"/>
      </rdf:Bag>
     </bqbiol:isVersionOf>
    </rdf:Description>
   </rdf:RDF>
  </annotation>
 </species>
 <species metaid="metaid_0000019" [..] />
 [..]
</listOfSpecies>
<listOfParameters>
 <parameter metaid="metaid_0000008" id="Pt" name="Total Per"
  constant="false"/>
 <parameter metaid="metaid_0000009" id="Tt" name="Total Tim"
  constant="false"/>
 <parameter metaid="metaid_0000010" id="V_mT" value="0.7">
  <notes>
   <body xmlns="http://www.w3.org/1999/xhtml">
    <p>V_mT=.7 (physiological oscillations); 0.28 (chaos); .4 or .99
       (biorhythmicity example)</p>
   </body>
  </notes>
 </parameter>
 <parameter metaid="metaid_0000011" id="V_dT" value="2">
  <notes>
   <body xmlns="http://www.w3.org/1999/xhtml">
    <p>V_dT=2 (physiological oscillations); 4.8 (chaos); 3.8 or 2
      (biorhythmicity example)</p>
   </body>
  </notes>
 </parameter>
</listOfParameters>
<listOfRules />
<listOfReactions>
 <reaction metaid="metaid_0000025" />
 <reaction metaid="metaid_0000026" id="T0_to_T1" name="r1"
  reversible="false">
  <annotation>
   <rdf:RDF>
    <rdf:Description rdf:about="#metaid_0000026">
     <bqbiol:isVersionOf>
      <rdf:Bag>
       <rdf:li rdf:resource="urn:miriam:ec-code:2.7.11.1"/>
       <rdf:li rdf:resource="urn:miriam:obo.go:GO\%3A0006468"/>
      </rdf:Bag>
     </bqbiol:isVersionOf>
    </rdf:Description>
   </rdf:RDF>
  </annotation>
```

```
     <listOfReactants>
      <speciesReference species="T0"/>
     </listOfReactants>
     <listOfProducts>
      <speciesReference species="T1"/>
     </listOfProducts>
     <kineticLaw />
    </reaction>
    <reaction metaid="metaid_0000027" [..] />
    [..]
    <reaction metaid="metaid_0000030" id="T1_to_T2" name="Second
        Phosphorylation of TIM" reversible="false">
     <listOfReactants>
      <speciesReference species="T1"/>
     </listOfReactants>
     <listOfProducts>
      <speciesReference species="T2"/>
     </listOfProducts>
     <kineticLaw />
    </reaction>
    <reaction metaid="metaid_0000031" [..] />
    <reaction metaid="metaid_0000032" [..] />
    <reaction metaid="metaid_0000033" [..] />
    <reaction metaid="metaid_0000034" [..] />
    <reaction metaid="metaid_0000035" [..] />
    <reaction metaid="metaid_0000036" [..] />
    <reaction metaid="metaid_0000038" id="T2_degradation" name="TIM-2
        degradation" reversible="false" sboTerm="SBO:0000179">
     <annotation>
      <rdf:RDF>
       <rdf:Description rdf:about="#metaid_0000038">
        <bqbiol:isVersionOf>
         <rdf:Bag>
          <rdf:li rdf:resource="urn:miriam:obo.go:GO\%3A0030163"/>
         </rdf:Bag>
        </bqbiol:isVersionOf>
       </rdf:Description>
      </rdf:RDF>
     </annotation>
     <listOfReactants>
      <speciesReference species="T2"/>
     </listOfReactants>
     <kineticLaw />
    </reaction>
    <reaction metaid="metaid_0000039" id="PT_complex_formation" name="PER-
        TIM complex formation">
     <annotation>
      <rdf:RDF>
       <rdf:Description rdf:about="#metaid_0000039">
        <bqbiol:isVersionOf>
         <rdf:Bag>
          <rdf:li rdf:resource="urn:miriam:obo.go:GO\%3A0006461"/>
         </rdf:Bag>
        </bqbiol:isVersionOf>
       </rdf:Description>
      </rdf:RDF>
     </annotation>
     <listOfReactants>
      <speciesReference species="P2"/>
      <speciesReference species="T2"/>
     </listOfReactants>
     <listOfProducts>
```

```
p01: Euglena() := new(end)((EuglenaLight(end) | EuglenaColl(end)))
p02: EuglenaLight(toColl) := (phot(). EuglenaLight(toColl) +
     phot().toColl<>. EuglenaUp() + phot(). toColl<>. EuglenaDown() +
     phot(). toColl<>. Euglena())
p03: EuglenaColl(toLight) := (coll<>. EuglenaColl(toLight) +
     toLight())
p04: EuglenaUp() := (coll(). Euglena() + coll<>. Euglena() +
     phot(). Euglena())
p05: EuglenaDown() := (coll(). Euglena() + coll<>. Euglena() +
     phot(). EuglenaDown())
p06: Photon() := phot<>.Photon()
i0:  (Photon()|(Euglena()))
```

Listing A.2: The Euglena model in process notation

```
     <speciesReference species="CC"/>
    </listOfProducts>
    <kineticLaw />   <reaction metaid="metaid_0000040" [..] />
   <reaction metaid="metaid_0000041" [..] />

   </reaction>
   <reaction metaid="metaid_0000040" [..] />
   <reaction metaid="metaid_0000041" [..] />
   <reaction metaid="metaid_0000042" [..] />
   <reaction metaid="metaid_0000043" [..] />
   <reaction metaid="metaid_0000044" [..] />
   <reaction metaid="metaid_0000045" [..] />
   <reaction metaid="metaid_0000046" [..] />
   <reaction metaid="metaid_0000047" [..] />
   <reaction metaid="metaid_0000048" [..] />
  </listOfReactions>
 </model>
</sbml>
```

Listing A.1: Modified SBML code of Leloup Goldbeter 1999

A.2.2. Euglena movements (πML)

Listing A.2 shows the definition of a simple model for Euglena movemens in dependency of light (using π Calculus notation). The model can be represented in πML. The according XML code is shown in Listing A.3.

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Euglena model -->
<pimodel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="euglenaSchemaBasic.xsd">
 <globalChannels>
  <channel name="coll" />
  <channel name="phot" />
 </globalChannels>
 <definitions>
  <!-- Euglena Prozess -->
  <definition name="Euglena">
   <new>
   <channel name="end" />
  </new>
  <process>
   <parallel>
    <process>
     <call name="EuglenaLight">
      <parameters>
       <channel name="end" />
      </parameters>
     </call>
    </process>
    <process>
     <call name="EuglenaColl">
      <parameters>
       <channel name="end" />
      </parameters>
     </call>
    </process>
   </parallel>
  </process>
 </definition>
 <!-- EuglenaLight Process -->
 <definition name="EuglenaLight">
  <parameters>
   <channel name="toColl" />
  </parameters>
  <process>
   <summation>
    <summand>
     <receive channel="phot" />
     <process>
      <call name="EuglenaLight">
       <parameters>
        <channel name="toColl" />
       </parameters>
      </call>
     </process>
    </summand>
    <summand>
     <receive channel="phot" />
     <process>
      <summation>
       <summand>
        <send channel="toColl" />
        <process>
         <call name="EuglenaUp" />
        </process>
       </summand>
      </summation>
     </process>
    </summand>
    <summand>
      <receive channel="phot" />
      <process>
       <summation>
        <summand>
         <send channel="toColl" />
         <process>
          <call name="EuglenaDown" />
         </process>
        </summand>
       </summation>
     </process>
    </summand>
    <summand>
     <receive channel="phot" />
     <process>
      <summation>
       <summand>
        <send channel="toColl" />
        <process>
         <call name="Euglena" />
        </process>
       </summand>
      </summation>
     </process>
    </summand>
   </summation>
  </process>
 </definition>
<!-- EuglenaColl Process -->
<definition name="EuglenaColl">
 <parameters>
  <channel name="toLight" />
 </parameters>
 <process>
  <summation>
   <summand>
    <send channel="coll" />
    <process>
     <call name="EuglenaColl">
      <parameters>
       <channel name="toLight" />
      </parameters>
     </call>
    </process>
   </summand>
   <summand>
    <receive channel="toLight" />
   </summand>
  </summation>
 </process>
</definition>
<!-- EuglenaUp Process -->
<definition name="EuglenaUp">
 <process>
  <summation>
   <summand>
    <receive channel="coll" />
    <process>
     <call name="Euglena" />
    </process>
   </summand>
   <summand>
    <send channel="coll" />
    <process>
     <call name="Euglena" />
    </process>
   </summand>
   <summand>
    <receive channel="phot" />
    <process>
     <call name="Euglena" />
    </process>
```

```
      </summand>
     </summation>
    </process>
   </definition>
   <!-- EuglenaDown Process -->
   <definition name="EuglenaDown">
    <process>
     <summation>
      <summand>
       <receive channel="coll" />
       <process>
        <call name="Euglena" />
       </process>
      </summand>
      <summand>
       <send channel="coll" />
       <process>
        <call name="Euglena" />
       </process>
      </summand>
      <summand>
       <receive channel="phot" />
       <process>
        <call name="Euglena" />
       </process>
      </summand>
     </summation>
    </process>
   </definition>
   <!-- Photon Process -->
   <definition name="Photon">
    <process>
     <summation>
      <summand>
       <send channel="phot" />
       <process>
        <call name="Photon" />
       </process>
      </summand>
     </summation>
    </process>
   </definition>
  </definitions>
  <!-- Init process definition sets the start conditions-->
  <initProcess>
   <call name="Photon" />
   <call name="Euglena" />
   <call name="Euglena" />
   <call name="Euglena" />
   <call name="Euglena" />
   <call name="Euglena" />
  </initProcess>
 </pimodel>
```

Listing A.3: A sample PiML model

A.3. Annotated bio-models

A.3.1. Example

The following example for the modeling and encoding of a biological question summarizes and illustrates the application of the different modeling and standardization efforts, using an enzymatic reaction. Biochemical systems are often described through networks of biochemical reactions. Example A.3.1 shows the encoding of an enzymatic reaction.

Example A.3.1 [Enzyme kinetic reaction with enzyme-substrate complex]

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_3} E + P$$

$E = 5 \times 10^{-21}\ mol$
$S = 10^{-20}\ mol$
$P = 0\ mol$
$ES = 0\ mol$

$k_1 = 0.0001\ l * mol^{-1} * second^{-1}$
$k_2 = 0.001 * second^{-1}$
$k_3 = 1 * second^{-1}$

It is a one-substrate reaction describing the reversible formation of an enzyme-substrate complex (ES) from an enzyme (E) and a substrate (S), and an irreversible reaction describing the degradation of the enzyme-substrate into the enzyme (E) and a product (P). The system provides initial amounts for all entities involved (e. g. the initial amount of the substrate S is $10^{-20}\ mol$). Furthermore, the reaction rates for all three reactions are given (e. g. the reaction rate for the formation of ES is $k_1 = 0.0001\ l/(mol * second)$).

In order to calculate the temporal behavior of the system, modeling and/or programming languages may be used, for example MatLab[3]. A sample *.m* file implementing parts of Example A.3.1 is shown in Listing A.4. The script defines a set of global parameters, corresponding to the kinetic rates in the example. It then sets the initial parametrization of the model at time 0 for the four modeled entities (E,S,P,ES) and finally calls an ODE solver ($ode45$). The system of differential equations can directly be derived from the system of chemical equations given in Listing A.3.1 but is irrelevant for the example and therefore omitted.

[3]`http://www.mathworks.com/products/matlab/`, last accessed 08 January 2011.

```
% Make rate constants available to subroutines
  global k1,k2,k3
% Read/assign model parameters
  k1 = 1e-4.;
  k2 = 1e-3.;
  k3 = 1.;
% Initial values
  y0(1)= 5e-21;
  y0(2)= 1e-20.;
  y0(3)= 0.;
  y0(4)= 0.;
% Call a routine to solve ODE
  [t, y]=ode45('michael', [0:0.1:10], y0);
```

Listing A.4: Extract from a MatLab script to solve the system provided in Example A.3.1. Parameter definitions and initial values are shown. Equations are omitted.

The known problems with code reuse in software design research also apply here: A model encoded in MatLab is only reusable in a particular software environment and hardly retrievable. Representation formats (Section 2.2) provide support for model export, import, and also storage. An SBML snippet representing part of the above example is given in Listing A.5. The SBML file encodes each biological entity as a separate `species` in the `listOfSpecies` (ll. 8-13). Each species has a `name` assigned to it, corresponding with the names given to the entities in Example A.3.1, and each species has a defined `initialConcentration` which corresponds with the amount given in Example A.3.1 (due to the fact that the model consistts only of one compartment of size 1).

```
1  <sbml>
2   <model metaid="COPASI1" id="Model_1" name="NoTitle">
3    <listOfUnitDefinitions> [..] </listOfUnitDefinitions>
4    <listOfCompartments>
5     <compartment id="compartment_1" name="compartment" size="1"/>
6    </listOfCompartments>
7    <listOfSpecies>
8     <species metaid="m001" id="species_1" name="E" compartment="
          compartment_1" initialConcentration="5e-21"/>
9     <species metaid="m002" id="species_2" name="S" compartment="
          compartment_1" initialConcentration="1e-20"/>
10    <species metaid="m003" id="species_3" name="ES" compartment="
          compartment_1" initialConcentration="0"/>
11    <species metaid="m004" id="species_4" name="P" compartment="
          compartment_1" initialConcentration="0"/>
12   </listOfSpecies>
13   <listOfReactions>
14    <reaction id="reaction_1" name="enzyme-substrate complex" reversible="
          true">
15     <listOfReactants>
16      <speciesReference species="species_1"/>
17      <speciesReference species="species_2"/>
18     </listOfReactants>
19     <listOfProducts>
```

```
20      <speciesReference species="species_3"/>
21     </listOfProducts>
22     <kineticLaw> [..]
23      <listOfParameters>
24       <parameter id="k1" name="k1" value="1e-4"/>
25       <parameter id="k2" name="k2" value="0.001"/>
26      </listOfParameters>
27     </kineticLaw>
28    </reaction>
29    <reaction id="reaction_2" name="product" reversible="false"> [..]
30    </reaction>
31   </listOfReactions>
32  </model>
33 </sbml>
```

Listing A.5: Enzymatic reaction in SBML notation

The model encodes one standard compartment `compartment_1` (l. 6). The chemical reactions shown in Example A.3.1 are encoded in the SBML `listOfReactions` element (ll. 14-34), each reaction in a single `reaction` element. The participating species are referenced from the `listOfReactants` (ll. 16-19 for `reaction_1`) or the `listOfProducts` (ll. 20-22), respectively. The reaction kinetics are encoded in the `kineticLaw` element where a `listOfParameters` defines a `parameter` for each kinetic rate (l. 25 specifies `k1=1e-4`).

The representation format is the basis for model storage and the later retrieval and ranking concept (Chapter 7). However, considering only the XML structure of the encoded model is not sufficient for a model retrieval. Therefore, additional meta-information can be attached to the SBML file using the SBO introduced in Section 3.1.2 and the MIRIAM annotation scheme introduced in Section 3.2.1. An annotated SBML species is shown in Listing A.6.

```
<sbml>
 <model metaid="COPASI1" id="Model_1" name="NoTitle"> [..]
  <listOfSpecies>
   <species metaid="m001" id="species_1" name="E" compartment="
        compartment_1" initialConcentration="5e-21">
    <annotation>
     <rdf:RDF ..>
      <rdf:Description rdf:about="#m001">
       <bqbiol:is>
        <rdf:Bag>
         <rdf:li rdf:resource="urn:miriam:uniprot:P00439" />
        </rdf:Bag>
       </bqbiol:is>
      </rdf:Description>
     </rdf:RDF>
    </annotation>
   </species>
     [..]
  </listOfSpecies>
  <listOfReactions>
   <reaction id="reaction_1" name="enzyme-substrate complex" [..]
    sbo="SBO:0000432" />
   [..]
  </listOfReactions>
 </model>
</sbml>
```

Listing A.6: Annotated SBML snippet

The given example also illustrates the distinction between data, meta-data, information and meta-information as discussed in Section 2.3: The *data* of the SBML file includes the values of the XML attributes (e. g. $reaction1$ or $5e-21$). The *meta-data* includes facts such as that $reaction1$ is a *reaction*, or that $5e-21$ is the initial concentration of a species ($initialConcentration$)[4]. The data and meta-data together allow to draw conclusions sufficient, for example, to simulate the model. However, *meta-information* provides additional information about the encoded mathematics and biology. Information on the mathematical basis of reaction $reaction1$, for example is provided by the `sbo` attribute (l. 21). The associated SBO term identifies the reaction as an *irreversible Michaelis Menten rate law for two substrates.* Information on the biological meaning of species $species1$, as another example for meta-information, is provided by the $< annotation >$ element (ll. 6-16). The annotation identifies the species as the protein *Phenylalanine-4-hydroxylase* as the URN `urn:miriam:uniprot:P00439` which points to the particular entry in the UniProt Database is connected with the species element through the qualifier $bqbiol : is$.

A.3.2. πML annotations

An example for an annotated version of the πML model presented in Appendix A.2.2 is given in Listing A.7.

```

<piml  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="euglenaSchemaBasic.xsd">
<!-- proposed annotation element -->
<annotation>
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:dcterms="http://purl.org/dc/terms/"
          xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#"
          xmlns:bqbiol="http://biomodels.net/biology-qualifiers/"
          xmlns:bqmodel="http://biomodels.net/model-qualifiers/">
 <rdf:Description rdf:about="#metaid_000001">
 <!-- information on model creator -->
 <dc:creator rdf:parseType="Resource">
   <rdf:Bag>
    <rdf:li rdf:parseType="Resource">
     <vCard:N rdf:parseType="Resource">
      <vCard:Given>Robert</vCard:Given>
       <vCard:Family>Kuehn</vCard:Family>
      </vCard:N>
      <vCard:EMAIL>robert.kuehn@uni-rostock.de</vCard:EMAIL>
      <vCard:ORG>
       <vCard:Orgname>University of Rostock</vCard:Orgname>
```

[4]Unit information are omitted in the SBML code snippet.

```
      </vCard:ORG>
     </rdf:li>
    </rdf:bag>
   </dc:creator>
  </rdf:Description>
  <rdf:Description rdf:about="#metaid_000005">
   <bqbiol:identity>
    <rdf:Bag>
     <rdf:li rdf:id="urn:miriam:taxonomy:3038" />
    </rdf:Bag>
   </bqbiol:is>
  </rdf:Description>
  <rdf:Description rdf:about="#metaid_000004">
   <bqbiol:is>
    <rdf:Bag>
     <rdf:li rdf:id="urn:miriam:obo.go:GO%3A0015979" />
     <rdf:li rdf:id="urn:miriam:obo.go:GO%3A0019684" />
    </rdf:Bag>
   </bqbiol:is>
  </rdf:Description>
 </rdf:RDF>
</annotation>

<!-- original PiML model -->
<pimodel metaid="metaid_000001">
 <globalChannels>
  <channel name="coll" metaid="metaid_000003" />
  <channel name="phot" metaid="metaid_000004" />
 </globalChannels>
 <definitions>
 <!-- Euglena Prozess -->
  <definition name="euglenaSpecies" metaid="metaid_000005">
   <new>
    <channel name="end" metaid="metaid_000006"/>
   </new>
   <process metaid="metaid_000007">
    <parallel>
     <process metaid="metaid_000008">
      <call name="EuglenaLight">
      <!-- defined further down -->
       <parameters>
        <channel name="end" />
       </parameters>
      </call>
     </process>
     <process metaid="metaid_000009">
      <call name="EuglenaColl">
       <!-- defined further down -->
       <parameters>
        <channel name="end" />
       </parameters>
      </call>
     </process>
    </parallel>
   </process>
  </definition> [..]
 </definitions>
 <initProcess metaid="metaid_000010">
  [..]
 </initProcess>
```

```
84 </pimodel>
```

Listing A.7: An annotated πML model snippet, using the proposed MIRIAM reference standard. The XML Schema for πML has been extended following the changes proposed in the textual description.

The lower part of the listing shows a sample process definition complying with the current πML XSD Schema (Euglena species). The upper part shows the possible annotation of that process and its channels. First, the model creator is defined (ll. 14-27). Then, the definition of `euglenaSpecies` is annotated with the NCBI Taxonomy term “Euglena” (`urn:miriam:taxonomy:3038`, ll. 29-35). Finally, the `phot` channel is annotated with the Gene Ontology term “photosynthesis, light reaction” (`urn:miriam:obo.go:GO%3A0015979` and `urn:miriam:obo.go:GO%3A0019684`, ll. 36-43). Further annotations might be added.

A.3.3. Goldbeter 1991 (CellML with MIRIAM annotations)

A proposal to use MIRIAM annotations in CellML has been posted on the cellml-team discussion mailing list, available from `http://www.cellml.org/pipermail/team/2009-August/001738.html`. The following encodes and annotates the model published in “A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase” [Goldbeter 1991] in CellML. The original annotated CellML model has been provided by Catherine Lloyd via e-mail exchange. The XML snippet shown in Listing A.8 is a shortened version of it. Namespaces are omitted. Left out code snippets are marked with “[..]”.

```
<model [..] cmeta:id="goldbeter_1991" name="goldbeter_1991">
 <documentation xmlns="http://cellml.org/tmp-documentation">
  [..]
 </documentation>
 <units name="minute"> [..] </units>

 <component cmeta:id="C" name="C">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <rdf:Description rdf:about="#C">
    <dc:title>C</dc:title>
    <dcterms:alternative>cyclin concentration</dcterms:alternative>
   </rdf:Description>
  </rdf:RDF>
  <variable units="micromolar" public_interface="out" cmeta:id="C_C" name
      ="C" initial_value="0.01">
   <rdf:RDF [..]>
    <rdf:Description rdf:about="#C_C">
     <bqbiol:isVersionOf>
      <rdf:Bag>
       <rdf:li rdf:resource="urn:miriam:uniprot:Q4KLA0"/>
       <rdf:li rdf:resource="urn:miriam:interpro:IPR006670"/>
       <rdf:li rdf:resource="urn:miriam:obo.sbo:sbo%3A0000252"/>
      </rdf:Bag>
     </bqbiol:isVersionOf>
    </rdf:Description>
   </rdf:RDF>
 </variable>
 <variable units="dimensionless" public_interface="in" name="X"/>
  [..]
 <variable units="minute" public_interface="in" name="time"/>
 <math xmlns="http://www.w3.org/1998/Math/MathML">
  [..]
 </math>
</component>

<component cmeta:id="M" name="M"> [..]  </component>
<component cmeta:id="M_star" name="M_star"> [..]  </component>
<component cmeta:id="X" name="X"> [..]  </component>

<component cmeta:id="X_star" name="X_star">
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="#X_star">
   <dc:title>X_star</dc:title>
   <dcterms:alternative>fraction of inactive cyclin protease</
       dcterms:alternative>
  </rdf:Description>
 </rdf:RDF>
 <variable units="dimensionless" public_interface="out" name="X_star"
     cmeta:id="X_X_">
  <rdf:RDF>
   <rdf:Description rdf:about="#X_X_">
    <bqbiol:isVersionOf>
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:obo.go:GO%3A0005680"/>
      <rdf:li rdf:resource="urn:miriam:obo.sbo:sbo%3A0000297"/>
     </rdf:Bag>
    </bqbiol:isVersionOf>
    <bqbiol:hasVersion>
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:reactome:REACT_7165.1"/>
     </rdf:Bag>
    </bqbiol:hasVersion>
   </rdf:Description>
  </rdf:RDF>
 </variable>
 <variable units="dimensionless" public_interface="in" cmeta:id="X_X"
     name="X"/>
  [..]
</component>

<component name="model_parameters"> [..]  </component>

<connection>
 <map_components component_2="C" component_1="environment"/>
 <map_variables variable_2="time" variable_1="time"/>
</connection>
<connection>
 <map_components component_2="M" component_1="environment"/>
 <map_variables variable_2="time" variable_1="time"/>
</connection>
 [..]
<connection>
 <map_components component_2="model_parameters" component_1="X"/>
 <map_variables variable_2="V3" variable_1="V3"/>
 <map_variables variable_2="V4" variable_1="V4"/>
 <map_variables variable_2="K3" variable_1="K3"/>
 <map_variables variable_2="K4" variable_1="K4"/>
</connection>
```

```

<rdf:RDF>
 <rdf:Description rdf:about="">
  <dc:creator rdf:parseType="Resource">
   <bqs:Person rdf:parseType="Resource">
    <vCard:N rdf:parseType="Resource">
     <vCard:Family>Lloyd</vCard:Family>
     <vCard:Given>Catherine</vCard:Given>
     <vCard:Other>May</vCard:Other>
    </vCard:N>
   </bqs:Person>
   <bqs:Person rdf:parseType="Resource">
     [..]
   </bqs:Person>
    [..]
  </dc:creator>
 </rdf:Description>
 <rdf:Description rdf:about="#goldbeter_1991">
  <bqs:reference rdf:parseType="Resource">
   <dc:subject rdf:parseType="Resource">
    <bqs:subject_type>keyword</bqs:subject_type>
    <rdf:value>
     <rdf:Bag>
      <rdf:li>oscillator</rdf:li>
      <rdf:li>cell cycle</rdf:li>
      <rdf:li>cyclin</rdf:li>
      <rdf:li>kinase</rdf:li>
     </rdf:Bag>
    </rdf:value>
   </dc:subject>
  </bqs:reference>
  <bqs:reference rdf:parseType="Resource">
   <bqs:Pubmed_id>1833774</bqs:Pubmed_id>
   <bqs:JournalArticle rdf:parseType="Resource">
    <dc:creator>
     <rdf:Seq>
      <rdf:li rdf:parseType="Resource">
       <bqs:Person rdf:parseType="Resource">
        <vCard:N rdf:parseType="Resource">
         <vCard:Family>Goldbeter</vCard:Family>
         <vCard:Given>A</vCard:Given>
        </vCard:N>
       </bqs:Person>
      </rdf:li>
     </rdf:Seq>
    </dc:creator>
    <dc:title>
     A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase
    </dc:title>
    <dcterms:issued rdf:parseType="Resource">
     <dcterms:W3CDTF>1991-10-15</dcterms:W3CDTF>
    </dcterms:issued>
    <bqs:Journal rdf:parseType="Resource">
     <dc:title>Proceedings of the National Academy of Sciences USA</dc:title>
    </bqs:Journal>
    <bqs:volume>88</bqs:volume>
    <bqs:first_page>9107</bqs:first_page>
    <bqs:last_page>9111</bqs:last_page>
   </bqs:JournalArticle>
  </bqs:reference>
```

```
    <bqmodel:isDescribedBy>
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:pubmed:1833774"/>
     </rdf:Bag>
    </bqmodel:isDescribedBy>
    <bqbiol:isHomologTo>
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:reactome:REACT_152"/>
     </rdf:Bag>
    </bqbiol:isHomologTo>
    <bqbiol:isVersionOf>
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:kegg.pathway:hsa04110"/>
      <rdf:li rdf:resource="urn:miriam:obo.go:GO%3A0000278"/>
     </rdf:Bag>
    </bqbiol:isVersionOf>
    <bqbiol:is>
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:taxonomy:8292"/>
     </rdf:Bag>
    </bqbiol:is>
   </rdf:Description>
  </rdf:RDF>
</model>
```

Listing A.8: Annotated CellML model

A.4. SED-ML

A.4.1. SED-ML language URNs

Table A.4 shows the standard URNs for reference to a particular model encoding language, and the corresponding URL for the language specification. The up-to-date list of SED-ML language URNs is availabe from `http://biomodels.net/sed-ml/#sedmlLanguage`.

Language	URN	Specification URL
CellML (generic)	`urn:sedml:language:cellML`	*none*
CellML 1.0	`urn:sedml:language:cellml.1_0`	`http://www.cellml.org/specifications/cellml_1.0`
CellML 1.1	`urn:sedml:language:cellml.1_1`	`http://www.cellml.org/specifications/cellml_1.1`
NeuroML (generic)	`urn:sedml:language:neuroml`	*none*
NeuroML Version 1.8.1 Level 1	`urn:sedml:language:neuroml.version-1_8_1.level-1`	*none*
NeuroML Version 1.8.1 Level 2	`urn:sedml:language:neuroml.version-1_8_1.level-2`	*none*
SBML (generic)	`urn:sedml:language:sbml`	*none*
SBML Level 1 Version 1	`urn:sedml:language:sbml.level-1.version-1`	`http://sbml.org/Special/specifications/sbml-level-1/version-1/html/`
SBML Level 1 Version 2	`urn:sedml:language:sbml.level-1.version-2`	`http://sbml.org/Special/specifications/sbml-level-1/version-2/html/`
SBML Level 2 Version 1	`urn:sedml:language:sbml.level-2.version-1`	`http://www.sbml.org/specifications/sbml-level-2/version-1/html/`

SBML Level 2 Version 2	urn:sedml:language:sbml.level-2.version-2	http://www.sbml.org/specifications/sbml-level-2/version-2/revision-1/sbml-level-2-version-2-rev1.pdf
SBML Level 2 Version 3	urn:sedml:language:sbml.level-2.version-3	http://www.sbml.org/specifications/sbml-level-2/version-3/release-1/sbml-level-2-version-3-rel-1.pdf
SBML Level 2 Version 3 (Release 1)	urn:sedml:language:sbml.level-2.version-3.release-1	http://www.sbml.org/specifications/sbml-level-2/version-3/release-1/sbml-level-2-version-3-rel-1.pdf
SBML Level 2 Version 3 (Release 2)	urn:sedml:language:sbml.level-2.version-3.release-2	http://precedings.nature.com/documents/58/version/2
SBML Level 2 Version 4	urn:sedml:language:sbml.level-2.version-4	http://precedings.nature.com/documents/2715/version/1
SBML Level 3 Version 1	urn:sedml:language:sbml.level-3.version-1	http://precedings.nature.com/documents/4123/version/1
VCML (generic)	urn:sedml:language:vcml	*none*

Table A.4.: SED-ML language URNs

A.4.2. The SED-ML XML Schema

The full SED-ML XML Schema is available from the Sourceforge SVN on `http://sed-ml.svn.sourceforge.net/`. The reference version for this Ph.D. work is the file `sed-ml-L1-V1.xsd`, revision 265. The published SED-ML Level 1 Version 1 XML Schema definition is shown in Listing A.9.

```
<xs:schema targetNamespace="http://www.biomodels.net/sed-ml"
    xmlns="http://www.biomodels.net/sed-ml" xmlns:xs="http://www.w3.org
        /2001/XMLSchema"
    xmlns:math="http://www.w3.org/1998/Math/MathML">
 <xs:import namespace="http://www.w3.org/1998/Math/MathML" schemaLocation
     ="sbml-mathml.xsd" />
 <xs:simpleType name="SId">
  <xs:annotation>
   <xs:documentation>
    The type SId is used throughout SED-ML as the type of the 'id'
        attributes on model elements.
   </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
   <xs:pattern value="(_|[a-z]|[A-Z])(_|[a-z]|[A-Z]|[0-9])*" />
  </xs:restriction>
 </xs:simpleType>

 <!--  attribute group for elements with ID/name att -->
 <xs:attributeGroup name="idGroup">
  <xs:attribute name="id" use="required" type="SId"></xs:attribute>
  <xs:attribute name="name" use="optional" type="xs:string"></
      xs:attribute>
 </xs:attributeGroup>

 <!-- SED Base class -->
 <xs:complexType name="SEDBase">
  <xs:annotation>
   <xs:documentation xml:lang="en">
    The SEDBase type is the base type of all main types in SED-ML. It
        serves as a container for the annotation of any part of the
        experiment description.
   </xs:documentation>
  </xs:annotation>
  <xs:sequence>
   <xs:element ref="notes" minOccurs="0" />
   <xs:element ref="annotation" minOccurs="0" />
  </xs:sequence>
  <!--  This must be a variable-type identifier ,i.e., (Letter | '_') (
      NCNameChar)* that is unique in the document. -->
  <xs:attribute name="metaid" type="xs:ID" use="optional"></xs:attribute>
 </xs:complexType>
 <xs:element name="sedML">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="SEDBase">
     <xs:sequence>
      <xs:element ref="listOfSimulations" minOccurs="0" />
      <xs:element ref="listOfModels" minOccurs="0" />
      <xs:element ref="listOfTasks" minOccurs="0" />
      <xs:element ref="listOfDataGenerators" minOccurs="0" />
      <xs:element ref="listOfOutputs" minOccurs="0" />
     </xs:sequence>
     <xs:attribute name="level" type="xs:decimal" use="required" fixed="1
        " />
     <xs:attribute name="version" type="xs:decimal" use="required" fixed=
        "1" />
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <!-- notes and annotations -->
 <xs:element name="notes">
  <xs:complexType>
   <xs:sequence>
    <xs:any namespace="http://www.w3.org/1999/xhtml" processContents="
       skip" minOccurs="0" maxOccurs="unbounded" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="annotation">
  <xs:complexType>
   <xs:sequence>
    <xs:any processContents="skip" minOccurs="0" maxOccurs="unbounded" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!-- KiSAO ID type -->
 <xs:simpleType name="KisaoType">
  <xs:restriction base="xs:string">
   <xs:pattern value="KISAO:[0-9][0-9][0-9][0-9][0-9][0-9][0-9]" />
  </xs:restriction>
 </xs:simpleType>

 <!-- global element declarations -->
 <xs:element name="variable">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="SEDBase">
    <!--  at least one of taskReference or modelReference must be set -->
     <xs:attribute name="taskReference" type="SId" use="optional" />
     <xs:attribute name="modelReference" type="SId" use="optional" />
     <!-- either target or symbol have to be used in the variable
        definition -->
     <xs:attribute name="target" type="xs:token" use="optional" />
    <xs:attribute name="symbol" type="xs:string" use="optional" />
    <xs:attributeGroup ref="idGroup" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="parameter">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:attributeGroup ref="idGroup" />
    <xs:attribute name="value" type="xs:double" use="required" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="algorithm">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:attribute name="kisaoID" type="KisaoType" use="required" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="uniformTimeCourse">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:sequence>
     <xs:element ref="algorithm" />
    </xs:sequence>
    <xs:attributeGroup ref="idGroup" />
    <xs:attribute name="outputStartTime" type="xs:double" use="required"
        />
    <xs:attribute name="outputEndTime" type="xs:double" use="required" /
       >
    <xs:attribute name="numberOfPoints" type="xs:integer" use="required"
        />
    <xs:attribute name="initialTime" type="xs:double" use="required" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="task">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:attribute name="simulationReference" type="SId" use="required" /
       >
    <xs:attribute name="modelReference" type="SId" use="required" />
    <xs:attributeGroup ref="idGroup" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="plot2D">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:sequence>
     <xs:element ref="listOfCurves" minOccurs="0" />
```

```
      </xs:sequence>
      <xs:attributeGroup ref="idGroup" />
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="plot3D">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="SEDBase">
      <xs:sequence>
       <xs:element ref="listOfSurfaces" minOccurs="0" />
      </xs:sequence>
      <xs:attributeGroup ref="idGroup" />
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="report">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="SEDBase">
      <xs:sequence>
       <xs:element ref="listOfDataSets" minOccurs="0" />
      </xs:sequence>
      <xs:attributeGroup ref="idGroup" />
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="model">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="SEDBase">
      <xs:sequence>
       <xs:element ref="listOfChanges" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="language" type="xs:anyURI" use="optional"
          default="urn:sedml:language:xml" />
      <xs:attribute name="source" type="xs:anyURI" use="required" />
      <xs:attributeGroup ref="idGroup" />
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <!-- math element, does not inherit from SEDBase -->
  <xs:element name="math" type="math:Math" />
  <!-- listOf elements -->
  <xs:element name="listOfVariables">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="SEDBase">
      <xs:sequence>
      <xs:element ref="variable" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="listOfParameters">
   <xs:complexType>
    <xs:complexContent>
    <xs:extension base="SEDBase">
     <xs:sequence>
      <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded" />
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="listOfTasks">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="SEDBase">
     <xs:sequence>
      <xs:element ref="task" minOccurs="0" maxOccurs="unbounded" />
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="listOfSimulations">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="SEDBase">
     <xs:sequence>
      <xs:element ref="uniformTimeCourse" minOccurs="0" maxOccurs="
          unbounded" />
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="listOfOutputs">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="SEDBase">
     <xs:sequence minOccurs="0">
      <xs:element ref="plot2D" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="plot3D" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="report" minOccurs="0" maxOccurs="unbounded" />
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="listOfModels">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="SEDBase">
     <xs:sequence>
      <xs:element ref="model" minOccurs="0" maxOccurs="unbounded" />
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="listOfDataGenerators">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="SEDBase">
     <xs:sequence>
      <xs:element ref="dataGenerator" minOccurs="0" maxOccurs="unbounded"
          />
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="listOfCurves">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="SEDBase">
      <xs:sequence>
       <xs:element ref="curve" maxOccurs="unbounded" />
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="listOfSurfaces">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="SEDBase">
      <xs:sequence>
       <xs:element ref="surface" maxOccurs="unbounded" />
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="listOfDataSets">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="SEDBase">
      <xs:sequence>
       <xs:element ref="dataSet" maxOccurs="unbounded" />
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <!-- change -->
  <xs:element name="listOfChanges">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="SEDBase">
      <xs:sequence>
       <xs:element ref="changeAttribute" minOccurs="0" maxOccurs="
           unbounded" />
       <xs:element ref="changeXML" minOccurs="0" maxOccurs="unbounded" />
       <xs:element ref="addXML" minOccurs="0" maxOccurs="unbounded" />
       <xs:element ref="removeXML" minOccurs="0" maxOccurs="unbounded" />
       <xs:element ref="computeChange" minOccurs="0" maxOccurs="unbounded"
            />
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="newXML">
   <xs:complexType>
    <xs:sequence>
     <xs:any processContents="skip" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
   </xs:complexType>
```

```
</xs:element>
<xs:element name="changeAttribute">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:attribute name="target" use="required" type="xs:token" />
    <xs:attribute name="newValue" type="xs:string" use="required" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="changeXML">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:sequence>
     <xs:element ref="newXML" />
    </xs:sequence>
    <xs:attribute name="target" use="required" type="xs:token" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="addXML">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:sequence>
     <xs:element ref="newXML" />
    </xs:sequence>
    <xs:attribute name="target" use="required" type="xs:token" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="removeXML">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:attribute name="target" use="required" type="xs:token" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<xs:element name="computeChange">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SEDBase">
    <xs:sequence>
     <xs:element ref="listOfVariables" minOccurs="0" />
     <xs:element ref="listOfParameters" minOccurs="0" />
     <xs:element ref="math" />
    </xs:sequence>
    <xs:attribute name="target" use="required" type="xs:token" />
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<!-- data generator -->
<xs:element name="dataGenerator">
 <xs:complexType>
  <xs:complexContent>
```

```
     <xs:extension base="SEDBase">
      <xs:sequence>
       <xs:element ref="listOfVariables" minOccurs="0" />
       <xs:element ref="listOfParameters" minOccurs="0" />
       <xs:element ref="math" />
      </xs:sequence>
      <xs:attributeGroup ref="idGroup" />
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="curve">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="SEDBase">
      <xs:attributeGroup ref="idGroup" />
       <xs:attribute name="yDataReference" type="SId" use="required" />
       <xs:attribute name="xDataReference" type="SId" use="required" />
       <xs:attribute name="logY" use="required" type="xs:boolean" />
       <xs:attribute name="logX" use="required" type="xs:boolean" />
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="surface">
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="SEDBase">
       <xs:attributeGroup ref="idGroup" />
       <xs:attribute name="yDataReference" type="SId" use="required" />
       <xs:attribute name="xDataReference" type="SId" use="required" />
       <xs:attribute name="zDataReference" type="SId" use="required" />
       <xs:attribute name="logY" use="required" type="xs:boolean" />
       <xs:attribute name="logX" use="required" type="xs:boolean" />
       <xs:attribute name="logZ" use="required" type="xs:boolean" />
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="dataSet">
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="SEDBase">
       <xs:attribute name="dataReference" type="SId" use="required"></
           xs:attribute>
       <xs:attribute name="label" use="required" type="xs:string" />
       <xs:attributeGroup ref="idGroup" />
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
  </xs:schema>
```

Listing A.9: SED-ML Level 1 Version 1 XML Schema

A.4.3. The SED-ML UML Schema

Figure A.1 shows the UML model for SED-ML Level 1 Version 1.

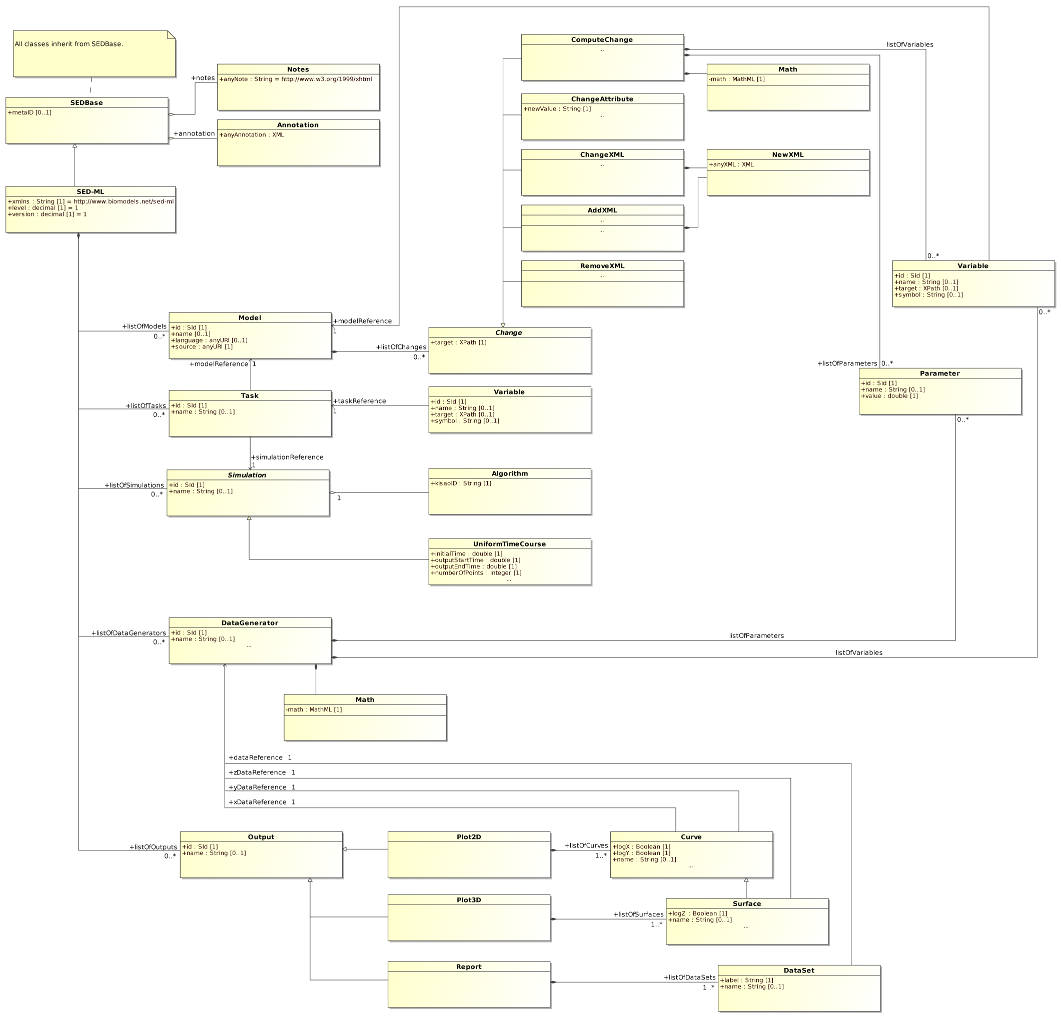


Figure A.1.: The SED-ML UML model.

A.4.4. Sample SED-ML experiment description

The following example is taken from [Waltemath et al. 2011b]. It provides a SED-ML description for the simulation of the model based on the publication by Leoup, Gonze and Goldbeter "Limit Cycle Models for Circadian Rhythms Based on Transcriptional Regulation in Drosophila and Neurospora" [Leloup et al. 1999]. The model source code is taken from the CellML Model Repository [Lloyd et al. 2008].

The model used in the simulation experiment is referred to with the URL (`http://models.cellml.org/workspace/leloup_gonze_goldbeter_1999/@@rawfile/d6613d7e1051b3eff2bb1d3d419a445bb8c754ad/leloup_gonze_goldbeter_1999_b.cellml`, ll. 15-16). In order to reproduce the simulation results given in the publication, the model needs to undergo some pre-processings. They are defined in the `listOfChanges` from ll. 17-25. Each defined change updates particular parameter values in the model.

A second model is defined in l. 28 in the example, using `model1` as a source and applying even further changes to it, in this case updating two more model parameters.

The simulation setup defined in the `listOfSimulations` is a `uniformTimeCourse` over 180 time units, using 1000 simulation points. The algorithm used is the CVODE solver, as denoted by the KiSAO ID `KiSAO:0000019`.

A number of `dataGenerator`s are defined in ll. 42-92. Those are the prerequisite for defining the output of the simulation. The first dataGenerator named `tim1` in l. 45 maps on the `Mt` entity in the model that is used in `task1` which here is the model with ID `model1`. The second dataGenerator named `per-tim` in l. 57 maps on the `CN` entity in `model1`. Finally the third and fourth dataGenerators map on the `Mt` and `per-tim` entity respectively in the updated model with ID `model2`.

The `output` defined in the experiment constists of a 2D plot with two different curves (ll. 96-102). Both curves plot the `per-tim` concentration against the `tim` concentration. In the first curve the original parametrisation (as given in `model1`) is used, in the second curve the updated one is used (as given in `model2`).

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <sedML version="0.1" xmlns="http://www.biomodels.net/sed-ml" xmlns:math="
       http://www.w3.org/1998/Math/MathML">
3   <!-- textual information about the experiment (optional) -->
4   <notes>Comparing Limit Cycles and strange attractors in Drosophila
5   </notes>
6   <!-- definition of simulation setup -->
7   <listOfSimulations>
8    <!-- uniform time course simulation with CVODE solver -->
9    <uniformTimeCourse id="simulation1" algorithm="KISAO:0000019"
         initialTime="0" outputStartTime="0" outputEndTime="180"
         numberOfPoints="1000" />
10  </listOfSimulations>
11
```

```
<!-- definition of models used during the experiment -->
<listOfModels>
 <!-- reference to a cellML model -->
 <model id="model1" name="Circadian Oscillations" type="CellML"
  source="http://models.cellml.org/workspace/leloup_gonze_goldbeter_1999
      /@@rawfile/d6613d7e1051b3eff2bb1d3d419a445bb8c754ad/
      leloup_gonze_goldbeter_1999_b.cellml" >
  <!-- changing initial conditions in the original model -->
  <listOfChanges>
   <changeAttribute target="/cellml:model/cellml:component[@cmeta:id='MP
       ']/cellml:variable[@name='vsP']/@initial_value" newValue="1"/>
   <changeAttribute target="/cellml:model/cellml:component[@cmeta:id='MP
       ']/cellml:variable[@name='vmP']/@initial_value" newValue="0.7"/>
   <changeAttribute target="/cellml:model/cellml:component[@cmeta:id='P2
       ']/cellml:variable[@name='vdP']/@initial_value" newValue="2"/>
   <changeAttribute target="/cellml:model/cellml:component[@cmeta:id='T2
       ']/cellml:variable[@name='vdT']/@initial_value" newValue="2"/>
   <changeAttribute target="/cellml:model/cellml:component[@name='
       parameters']/cellml:variable[@name='k1']/@initial_value" newValue
       ="0.6"/>
   <changeAttribute target="/cellml:model/cellml:component[@name='
       parameters']/cellml:variable[@name='K4P']/@initial_value"
       newValue="1"/>
   <changeAttribute target="/cellml:model/cellml:component[@name='
       parameters']/cellml:variable[@name='K4T']/@initial_value"
       newValue="1"/>
  </listOfChanges>
 </model>
 <!-- reference to the above model (model1) with additional changes of
     initial values of MY and T2 -->
 <model id="model2" name="Circadian Chaos" type="CellML" source="model1"
     >
  <listOfChanges>
   <changeAttribute target="/cellml:model/cellml:component[@cmeta:id='MT
       ']/cellml:variable[@name='vmT']/@initial_value" newValue="0.28"/>
   <changeAttribute target="/cellml:model/cellml:component[@cmeta:id='T2
       ']/cellml:variable[@name='vdT']/@initial_value" newValue="4.8"/>
  </listOfChanges>
 </model>
</listOfModels>
<!-- definition of tasks (combining simulation setup and model) -->
<listOfTasks>
 <!-- limit cycle on model1 -->
 <task id="task1" name="Limit Cycle" modelReference="model1"
     simulationReference="simulation1"/>
 <!-- strange attractors on the further perturbated model model2 -->
 <task id="task2" name="Strange attractors" modelReference="model2"
     simulationReference="simulation1"/>
</listOfTasks>
<!-- definition of the data generators needed to produce the output -->
<listOfDataGenerators>
 <!-- definition of data generator for tim mRNA -->
 <dataGenerator id="tim1" name="tim mRNA">
  <listOfVariables>
   <variable id="v1" taskReference="task1" target="/cellml:model/
       cellml:component[@cmeta:id='MT']" />
  </listOfVariables>
  <math:math>
   <math:apply>
    <math:plus />
    <math:ci>v1</math:ci>
   </math:apply>
   </math:math>
  </dataGenerator>
  <!-- definition of data generator for the nuclear PER-TIM complex -->
  <dataGenerator id="per-tim" name="nuclear PER-TIM complex">
   <listOfVariables>
    <variable id="v1" taskReference="task1" target="/cellml:model/
        cellml:component[@cmeta:id='CN']" />
   </listOfVariables>
   <math:math>
    <math:apply>
     <math:plus />
     <math:ci>v1</math:ci>
    </math:apply>
   </math:math>
  </dataGenerator>
  <!-- definition of data generator for pertubated tim mRNA -->
  <dataGenerator id="tim2" name="tim mRNA (changed parameters)">
   <listOfVariables>
    <variable id="v2" taskReference="task2" target="/cellml:model/
        cellml:component[@cmeta:id='MT']" />
   </listOfVariables>
   <math:math>
    <math:apply>
     <math:plus />
     <math:ci>v2</math:ci>
    </math:apply>
   </math:math>
  </dataGenerator>
  <!-- definition of data generator for perturbated nuclear PER-TIM
      complex -->
  <dataGenerator id="per-tim2" name="nuclear PER-TIM complex">
   <listOfVariables>
    <variable id="v1" taskReference="task2" target="/cellml:model/
        cellml:component[@cmeta:id='CN']" />
   </listOfVariables>
   <math:math>
    <math:apply>
     <math:plus />
     <math:ci>v1</math:ci>
    </math:apply>
   </math:math>
  </dataGenerator>
 </listOfDataGenerators>
 <!-- output definition -->
  <listOfOutputs>
   <!-- definition of a 2D plot to show the tim mRNA concentration with
       different initial conditions -->
   <plot2D id="plot1" name="tim mRNA with Oscillation and Chaos">
    <!-- definition of two output curves, both plotting per-tim (original
        and perturbated) against the tim concentration (original and
        perturbated) -->
    <listOfCurves>
     <curve logX="false" logY="false" xDataReference="per-tim"
         yDataReference="tim1" />
     <curve logX="false" logY="false" xDataReference="per-tim2"
         yDataReference="tim2" />
    </listOfCurves>
   </plot2D>
  </listOfOutputs>
</sedML>
```

Listing A.10: LeLoup Model Simulation Description in SED-ML

A.5. mDB

A.5.1. Internal Representation format (XML Schema)

The following listing shows the XML Schema of the internal representation format used for the model import in *mDB* and for the exchange of information and meta-information in the *Sombi* framework[5].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- created by Robert Haelke (University Rostock) -->
<!-- Model xml schema for parser internal representation -->
<!-- Date: 2009-07-08 -->
<!-- extended by KSWS project "Sombi" (University Rostock) -->
<!-- Date: 2010-07-21 -->
<!-- changed by Dagmar Waltemath (University Rostock) -->
<!-- Date: 2010-08-27 -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <!-- model definition -->
 <xs:element name="model">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="nameID" type="xs:string" maxOccurs="1" minOccurs="1" />
    <xs:element name="name" type="xs:string" maxOccurs="1" minOccurs="0"/>
    <xs:element name="dbModelID" type="xs:string" maxOccurs="1" minOccurs="0"/>
    <xs:element name="dateCreated" type="xs:date" maxOccurs="1" minOccurs="0"/>
    <xs:element name="dateModified" type="xs:date" maxOccurs="1" minOccurs="0"/>
    <xs:element name="stable" type="xs:boolean" maxOccurs="1" minOccurs="0"/>
    <xs:element name="lastVersion" type="xs:boolean" maxOccurs="1" minOccurs="0"/>
    <xs:element name="modelDescription" type="xs:string" maxOccurs="1" minOccurs="0"/>
    <xs:element ref="modelFormalism" maxOccurs="1" minOccurs="1" />
    <xs:element ref="modelFile" maxOccurs="1" minOccurs="1" />
    <xs:element ref="listOfQualifiers" maxOccurs="1" minOccurs="0" />
    <xs:element ref="listOfKeywords" maxOccurs="1" minOccurs="0" />
    <xs:element ref="listOfAuthors" maxOccurs="1" minOccurs="0"/>
    <xs:element ref="listOfReferenceDescriptions" maxOccurs="1" minOccurs="0"/>
    <xs:element ref="listOfDistributionStatements" maxOccurs="1" minOccurs="0"/>
    <xs:element ref="listOfComponents" maxOccurs="1" minOccurs="0"/>
    <xs:element ref="listOfCompartments" maxOccurs="1" minOccurs="0"/>
    <xs:element ref="listOfSpecies" maxOccurs="1" minOccurs="0"/>
    <xs:element ref="listOfReactions" maxOccurs="1" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <!-- modelFormalism definition -->
 <xs:element name="modelFormalism">
  <xs:complexType>
```

[5]Last accessed 27 August 2010.

```
  <xs:sequence>
   <xs:element name="type" type="formalismURN" maxOccurs="1" minOccurs="1" />
  </xs:sequence>
 </xs:complexType>
</xs:element>

<!-- formalismURN definition -->
<xs:simpleType name="formalismURN">
<!-- reused from SED-ML language definitions -->
 <xs:restriction base="xs:anyURI">
  <xs:enumeration value="urn:sedml:language:cellML"/>
  <xs:enumeration value="urn:sedml:language:cellml.1_0"/>
  <xs:enumeration value="urn:sedml:language:cellml.1_1"/>
  <xs:enumeration value="urn:sedml:language:neuroml"/>
  <xs:enumeration value="urn:sedml:language:neuroml.version-1_8_1.level-1"/>
  <xs:enumeration value="urn:sedml:language:neuroml.version-1_8_1.level-2"/>
  <xs:enumeration value="urn:sedml:language:sbml"/>
  <xs:enumeration value="urn:sedml:language:sbml.level-1.version-1"/>
  <xs:enumeration value="urn:sedml:language:sbml.level-1.version-2"/>
  <xs:enumeration value="urn:sedml:language:sbml.level-2.version-1"/>
  <xs:enumeration value="urn:sedml:language:sbml.level-2.version-2"/>
  <xs:enumeration value="urn:sedml:language:sbml.level-2.version-3.release-1"/>
  <xs:enumeration value="urn:sedml:language:sbml.level-2.version-3.release-2"/>
  <xs:enumeration value="urn:sedml:language:sbml.level-2.version-4"/>
  <xs:enumeration value="urn:sedml:language:sbml.level-3.version-1"/>
  <xs:enumeration value="urn:sedml:language:vcml"/>
  <xs:enumeration value="urn:sedml:other" />
 </xs:restriction>
</xs:simpleType>

<!-- modelFile definition -->
<xs:element name="modelFile">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="fileName" type="xs:string"/>
   <xs:element name="fileVersion" type="xs:string" maxOccurs="1" minOccurs="0"/>
   <xs:element name="modelFile" type="xs:string" maxOccurs="1" minOccurs="0"/> <!-- the model file -->
   <xs:element name="annotationFile" type="xs:string" maxOccurs="1" minOccurs="0"/> <!-- the annotation file  -->
   <xs:element ref="experiment" maxOccurs="unbounded" minOccurs="0" />
  </xs:sequence>
 </xs:complexType>
</xs:element>

<!-- experiment definition -->
<xs:element name="experiment">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="experimentURL" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
   <xs:element name="experimentType" type="xs:string" maxOccurs="1" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

```

<!-- listOfQualifiers defintion -->
<xs:element name="listOfQualifiers">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="modelQualifier" type="modelQualifierType" maxOccurs
       ="unbounded" minOccurs="1" >
   <!-- no two identical URIs within "modelQualifier" -->
    <xs:unique name="modelQualifierConstraint">
     <xs:selector xpath="URI" />
     <xs:field xpath="." />
    </xs:unique>
   </xs:element>
   <xs:element name="bioQualifier" type="bioQualifierType" maxOccurs="
       unbounded" minOccurs="1" >
    <!-- no two identical URIs within "bioQualifier" -->
    <xs:unique name="bioQualifierConstraint">
     <xs:selector xpath="URI" />
     <xs:field xpath="." />
    </xs:unique>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
 <!-- no two "modelQualifier" with same attribute -->
 <xs:unique name="modelQualifierConstraint2">
  <xs:selector xpath="modelQualifier" />
  <xs:field xpath="@modelQualifierRole" />
 </xs:unique>
 <!-- no two "bioQualifier" with same attribute -->
 <xs:unique name="bioQualifierConstraint2">
  <xs:selector xpath="bioQualifier" />
  <xs:field xpath="@bioQualifierRole" />
 </xs:unique>
</xs:element>

<!-- modelQualifier definition -->
<xs:complexType name="modelQualifierType">
 <xs:sequence>
  <xs:element name="URI" type="xs:anyURI" minOccurs="0" maxOccurs="
      unbounded" />
 </xs:sequence>
 <xs:attribute name="modelQualifierRole" use="required">
  <xs:simpleType>
   <xs:restriction base="xs:string">
    <xs:enumeration value="is"/>  <!-- = "modelURI" -->
    <xs:enumeration value="isDescribedBy"/>  <!-- = "pubmed" -->
    <xs:enumeration value="isDerivedFrom"/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
</xs:complexType>


<!-- bioQualifier definition -->
<xs:complexType name="bioQualifierType">
 <xs:sequence>
  <xs:element name="URI" type="xs:anyURI" minOccurs="0" maxOccurs="
      unbounded" />
 </xs:sequence>
 <xs:attribute name="bioQualifierRole" use="required">
  <xs:simpleType>
   <xs:restriction base="xs:string">
```

```
      <xs:enumeration value="encodes"/>
      <xs:enumeration value="hasPart"/>
      <xs:enumeration value="hasProperty"/>
      <xs:enumeration value="hasVersion"/>
      <xs:enumeration value="is"/>
      <xs:enumeration value="isDescribedBy"/>
      <xs:enumeration value="isEncodedBy"/>
      <xs:enumeration value="isHomologTo"/>
      <xs:enumeration value="isPartOf"/>
      <xs:enumeration value="isPropertyOf"/>
      <xs:enumeration value="isVersionOf"/>
      <xs:enumeration value="occursIn"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
  </xs:complexType>

  <!-- listOfKeywords definition -->
  <xs:element name="listOfKeywords">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="keyword" type="xs:string" maxOccurs="unbounded"
         minOccurs="0"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>

  <!-- listOfAuthors definition -->
  <xs:element name="listOfAuthors">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="author" type="authorType" maxOccurs="unbounded"
         minOccurs="1"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>

  <!-- author definition -->
  <xs:complexType name="authorType">
   <xs:sequence>
    <xs:element name="firstName" type="xs:string" maxOccurs="1" minOccurs=
        "1"/>
    <xs:element name="lastName" type="xs:string" maxOccurs="1" minOccurs="
        1"/>
    <xs:element name="organisation" type="xs:string" maxOccurs="1"
        minOccurs="0"/>
    <xs:element name="email" type="xs:anyURI" maxOccurs="1" minOccurs="0"/
        >
   </xs:sequence>
   <xs:attribute name="authorRole">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="submitter"/>
      <xs:enumeration value="creator"/>
      <xs:enumeration value="adder"/> <!-- wrote annotation -->
      <xs:enumeration value="publisher"/> <!-- wrote referenceDescription
          -->
      <xs:enumeration value="other"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
  </xs:complexType>
```

```

<!-- listOfReferenceDescriptions definition -->
<xs:element name="listOfReferenceDescriptions">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="referenceDescription" type="refDescType"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<!-- referenceDescription definition -->
<xs:complexType name="refDescType">
 <xs:sequence>
  <xs:element name="referenceURI" type="xs:anyURI" maxOccurs="1"
      minOccurs="0"/>
  <xs:element name="referenceTitle" type="xs:anyURI" maxOccurs="1"
      minOccurs="0"/>
  <xs:element ref="listOfAuthors" maxOccurs="1" minOccurs="0"/> <!-- or
      the Author himself? -->
  <xs:element name="referenceAbstract" type="xs:string" maxOccurs="1"
      minOccurs="0"/> <!-- just abstract, full text not available -->
 </xs:sequence>
</xs:complexType>

<!-- listOfDistributionStatements definition -->
<xs:element name="listOfDistributionStatements">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="distributionStatement" type="distStateType"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<!-- distributionStatement definition -->
<xs:complexType name="distStateType">
 <xs:sequence>
  <xs:element name="referenceURL" type="xs:anyURI" maxOccurs="1"
      minOccurs="0"/>
  <xs:element name="statementText" type="xs:string" maxOccurs="1"
      minOccurs="0"/>
 </xs:sequence>
 <xs:attribute name="type" use="required">
  <xs:simpleType>
   <xs:restriction base="xs:string">
    <xs:enumeration value="gnuGPL"/>
    <xs:enumeration value="gnuLGPL"/>
    <xs:enumeration value="creativeCommons"/>
    <xs:enumeration value="BSD"/>
    <xs:enumeration value="otherLicense"/>  <!-- if not one of the first
        four -->
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
</xs:complexType>

<!-- basicComponentType definition -->
<xs:complexType name="basicComponentType">
 <xs:sequence>
  <xs:element name="id" type="xs:string"/>
  <xs:element name="metaid" type="xs:string"/>
  <xs:element name="name" type="xs:string" minOccurs="0"/>
```

```
   <xs:element name="notes" type="xs:string" minOccurs="0" maxOccurs="1"
       />
   <xs:element name="bioQualifier" type="bioQualifierType" maxOccurs="
       unbounded" minOccurs="0">
    <xs:unique name="bioQualifierConstraint3">
     <xs:selector xpath="URI" />
     <xs:field xpath="." />
    </xs:unique>
   </xs:element>
  </xs:sequence>
 </xs:complexType>

 <!-- listOfComponents definition -->
 <xs:element name="listOfComponents">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="component" type="componentType">
     <xs:unique name="bioQualifierConstraint4">
      <xs:selector xpath="bioQualifier" />
      <xs:field xpath="@bioQualifierRole" />
     </xs:unique>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <!-- component definition -->
 <xs:complexType name="componentType">
  <xs:complexContent>
   <xs:extension base="basicComponentType">
    <xs:sequence>
     <xs:element name="role" minOccurs="1" maxOccurs="1">
      <xs:simpleType>
       <xs:restriction base="xs:string">
    <xs:enumeration value="parameter"/>
    <xs:enumeration value="event"/>
    <xs:enumeration value="other"/>
       </xs:restriction>
      </xs:simpleType>
     </xs:element>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>

  <!-- listOfCompartments definition -->
  <xs:element name="listOfCompartments">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="compartment" type="basicComponentType">
      <xs:unique name="bioQualifierConstraint5">
       <xs:selector xpath="bioQualifier" />
       <xs:field xpath="@bioQualifierRole" />
      </xs:unique>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>

  <!-- listOfSpecies definition -->
  <xs:element name="listOfSpecies">
   <xs:complexType>
```

```
    <xs:sequence>
     <xs:element name="species" type="speciesType" maxOccurs="unbounded"
         minOccurs="0">
      <xs:unique name="bioQualifierConstraint6">
       <xs:selector xpath="bioQualifier" />
       <xs:field xpath="@bioQualifierRole" />
      </xs:unique>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>

  <!-- species definition -->
  <xs:complexType name="speciesType">
   <xs:complexContent>
    <xs:extension base="basicComponentType">
     <xs:sequence>
      <xs:element name="compartment" type="xs:string" minOccurs="0"/>
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>

  <!-- listOfReactions definition -->
  <xs:element name="listOfReactions">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="reaction" type="reactionType" maxOccurs="unbounded
         " minOccurs="0">
      <xs:unique name="bioQualifierConstraint7">
       <xs:selector xpath="bioQualifier" />
       <xs:field xpath="@bioQualifierRole" />
      </xs:unique>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>

 <!-- reaction definition -->
 <xs:complexType name="reactionType">
  <xs:complexContent>
   <xs:extension base="basicComponentType">
    <xs:sequence>
     <xs:element name="listOfReactants" type="speciesReferenceType"
         maxOccurs="1" minOccurs="0"/>
     <xs:element name="listOfProducts" type="speciesReferenceType"
         maxOccurs="1" minOccurs="0"/>
     <xs:element name="listOfModifiers" type="speciesReferenceType"
         maxOccurs="1" minOccurs="0"/>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>

 <!-- speciesReferenceType definition -->
 <xs:complexType name="speciesReferenceType">
  <xs:sequence>
   <xs:element name="speciesReference" type="xs:string" maxOccurs="
       unbounded" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
```

RF concept		IRF	*mDB*
SBML	πML		
			modelName
sbml:level/version	*namespace*	-	formalism:version
		-	formalism:name
		modelFormalism:type	formalism:referenceURN
		model:name	model:modelName
file name	*file name*	modelFile:fileName	-
implicit/user	*user*	model:stable	model:stable
-	-	-	model:lastVersion
-	-	modelFile:fileVersion	file:versionFile
model.xml	*model.xml*	modelFile:modelFile	file:modelFile
-	*annotation.xml*	modelFile:annotationFile	file:annotationFile

Table A.5.: Mapping of the different concepts in the administrative data dimension on the internal representation format (IRF) elements and on the *mDB* attributes, using the example of SBML and πML. The notation *xx:yy[@zz]* points to *element:attribute[@value]*; the notation *xx:yy* points to the *element:subelement* in the IRF column. The notation *relation:attribute* points to the element:attribute in the *mDB* column.

```
</xs:schema>
```

Listing A.11: Internal Representation Format (IRF) XML Schema

A.5.2. Representation format to IRF to mDB mapping

The following provides the mapping of the SBML and πML language elements on the IRF and *mDB*, respectively. The types of meta-information and data have been separated into the different *dimensions* introduced in [Henkel et al. 2010] and described in Section 7.3.1. Each of the dimensions, and the related mappings, are described in the following paragraphs.

Administrative data dimension The concepts relating to the administrative data dimension are shown in Table A.5.

The namespace used in the instances of the representation formats allow for the mapping on standard language URNs. For example, the SBML `level` and `version` attributes such as <sbml level="2" version="1" [..] /> can be resolved into the language URN `urn:sedml:language:sbml.level-2.version-1` that is kept in the IRF. For this URN the `name` "SBML" and the `version` "Level 2 Version 1" are stored in *mDB*.

The `stable` attribute in the IRF (and *mDB*) indicates whether the stored version of the model is a stable one, or being worked on (unstable). For all models imported from the curated branch of BioModels Database and CellML Model Repository the attribute is set to `stable=true`. Models imported from the non-curated branch of

RF concept		IRF	*mDB*
SBML	πML		
vcard	*vcard*	author:organisation	person:organisation
		author:lastName	person:lastName
		author:firstName	person:firstName
		author:email	person:email
user	*user*	author:authorRole[@submitter]	submittedBy
dc:creator	dc:creator	author:authorRole[@creator]	createdBy
from ref pub	*from ref pub*	author:authorRole[@publisher]	publishedBy
user	*user*	author:authorRole[@adder]	annotatedBy
user	*user*	author:authorRole[@main]	createdBy:main
from ref pub	*from ref pub*	author:authorRole[@main]	publishedBy:main
user	*user*	author:authorRole[@main]	annotatedBy:main

Table A.6.: Mapping of the different concepts in the person dimension on the internal representation format (IRF) elements and on the *mDB* attributes, using the example of SBML and πML. The notation *xx:yy[@zz]* points to *element:attribute[@value]*; the notation *xx:yy* points to the *element:subelement* in the IRF column. The notation *relation:attribute* points to the element:attribute in the *mDB* column.

BioModels Database have the attribute set to `stable=false`. For πML models, the information must be provided by the user.

The `lastVersion` attribute contains the `idmodel` of the last version of that model, i. e. if a newer version of a model is stored, the `lastVersion` attribute has to be updated on all previous versions of that model. The initial value is NULL, denoting the initial import of a model into the system. As the IRF only imports a model at a time, without any knowledge of the model's relatives, the `lastVersion` value is not stored in the IRF; the relation of a model to other existing models in *mDB* has to be indicated by the user.

The `versionFile` stores the version information for a particular model. It is used by the external versioning system (Section 6.4) to maintain the version tree. The `modelFile` stores the original model file. The `annotationFile` stores the original annotation file, if that is kept seperated from the model file, as for example offered by the πML language.

Person dimension The concepts relating to the person dimension are shown in Table A.6.

mDB has a general `person` relation that stores information on persons, including the organization, last name, first name and email. This construct corresponds to the `author` element in the IRF. The information can be extracted from the `vcard` elements in both SBML and πML.

To assign particular roles to the author, the `authorRole` in the IRF defines `submitter` (i. e. the person who submitted the model to *mDB*, not encoded in the

RF concept		**IRF**	***mDB***
SBML	πML		
dbms	*dbms*	model:dateSubmitted	modelAnnotation:submissionDate
dc:created	dc:created	model:dateCreated	modelAnnotation:creationDate
dc:modified	dc:modified	model:dateModified	modelAnnotation:modificationDate

Table A.7.: Mapping of the different concepts in the dates dimension on the internal representation format (IRF) elements and on the *mDB* attributes, using the example of SBML and πML. The notation *xx:yy[@zz]* points to *element:attribute[@value]*; the notation *xx:yy* points to the *element:subelement* in the IRF column. The notation *relation:attribute* points to the element:attribute in the *mDB* column.

RF), `creator` (i. e. the person who created the model, encoded in the `dc:creator` element of the RF), `publisher` (i. e. the persons who published the reference publication), and the `adder` (i. e. the person who encoded the annotations, often overlapping with the creator). For SBML, the publisher are determined by resolving the reference publication URN, and determining the authors of that publication, if accessible.

To assign a main creator to a model, as well as a main publisher, the `main` attribute is introduced. The information on the main creator has to be provided by the user. The main publisher usually is the first author of the reference publication. The authors are extracted from the reference publication by resolving the corresponding URN to the publication library entry given in the model annotation (Table A.8).

Dates dimension The concepts relating to the dates dimension are shown in Table A.7.

The `submissionDate` in *mDB* (`dateSubmitted` in IRF) refers to the time when the model was stored in *mDB*. The `creationDate` (`dateCreated` in IRF) refers to the time when the model representation file was initially created. That information is extracted from the `dc:created` element in the SBML model, and πML model respectively. The `modificationDate` (`dateModified` in IRF) refers to the time when the model representation file was last modified. That information is extracted from the `dc:modified` element in the SBML model, and πML model respectively. The full history of the model is, however, only traceable from the versioning system, as introduced in Section 6.4.

Publication dimension The concepts relating to the publication dimension are shown in Table A.8. If existing, the reference publication pointing to the original description of the model is given in the representation format, such as in the model annotation of an SBML file. That URN is stored in the IRF in the `referenceURI` element of the `referenceDescription`. It is then imported in *mDB*

RF concept		IRF	*mDB*
SBML	πML		
bqmodel: isDescribedBy	bqmodel: isDescribedBy	referenceDescription: referenceURI	publication: referenceURN
from ref pub	*from ref pub*	referenceDescription: referenceTitle	publication:title
from ref pub	*from ref pub*	referenceDescription: referenceAbstract	-
user	*user*		publication: descriptionText
		distributionStatement[@type]	distribution:type
user	*user*	distributionStatement: referenceURL	distribution: referenceURL
user	*user*	distributionStatement: statementText	distribution: statementText

Table A.8.: Mapping of the different concepts in the publication dimension on the internal representation format (IRF) elements and on the *mDB* attributes, using the example of SBML and πML. The notation *xx:yy[@zz]* points to *element:attribute[@value]*; the notation *xx:yy* points to the *element:subelement* in the IRF column. The notation *relation:attribute* points to the element:attribute in the *mDB* column.

into the `publication` relation, and there stored in the `referenceURN` attribute. Each URN is resolved to the corresponding web resource, and the information on the reference publication is stored in the `referenceTitle` element of the IRF and the `title` attribute of the `publication` relation respectively. Furthermore, the paper's abstract is stored in the `referenceAbstract` element of the IRF. That information is used during the index building process for the ranked retrieval (Chapter 7), but not provided publicly.

If no published reference publication exists for a model, the user may add a free-text description of the model, which is stored in the *mDB* in the `descriptionText` attribute of the `publication` relation.

The publication has a number of authors assigned through the `publishedBy` relation shown in Table A.6.

The distribution regulations are stored in the `distribution` relation of *mDB* for each model. The user can assign a predefined distribution type to the model (e. g. "gnuGPL" in the `type` attribute). In addition, the reference URL to the distribution statement definition (`referenceURL`) and the textual description of it are stored, to facilitate the definition of further statements (`statementText`), and to keep track on possible changes. The predefined types given by the IRF are listed in the XML Schema in Appendix A.5.1, ll. 241-251. All information on the distribution has to be provided by the user. However, if a model repository defines its distribution regulations, those are automatically taken over, as for example when importing the SBML models from BioModels Database. The statement text for public licenses

RF concept		IRF	*mDB*
SBML	πML		
		bioQualifier	
user	*user*	[@bioQualifierRole]	qualifier:name
		modelQualifier	
user	*user*	[@modelQualifierRole]	qualifier:name
user	*user*	-	qualifier:namespace
bqmodel:*any*	bqmodel:*any*	model:	modelOntologyEntry:
		modelQualifier:URI	modelOntologyURI
bqbiol:*any*	bqbiol*any*	model:	modelOntologyEntry:
		bioQualifier:URI	modelOntologyURI
bqbiol:*any*	bqbiol:*any*	bioQualifier:URI	componentOntologyEntry:
			bioOntologyURI
		component:name	component:name
		component:metaid	component:metaID
		component:id	component:
			modelComponentID
		species, reaction,	
		compartment definition	component[@role]
		role[@parameter]	component[@role]
		role[@event]	component[@role]
		role[@other]	component[@role]
		species:compartment	isLocatedIn
		reaction:listOfReactants:	participatesIn
		speciesReference	[@role=reactant]
		reaction:listOfProducts:	participatesIn
		speciesReference	[@role=product]
		reaction:listOfModifiers:	participatesIn
		speciesReference	[@role=modifier]

Table A.9.: Mapping of the different concepts in the constituents dimension on the internal representation format (IRF) elements and on the *mDB* attributes, using the example of SBML and πML. The notation *xx:yy[@zz]* points to *element:attribute[@value]*; the notation *xx:yy* points to the *element:subelement* in the IRF column. The notation *relation:attribute* points to the element:attribute in the *mDB* column.

is copied from the corresponding license definition, but own licenses may also be defined.

Constituents dimension The concepts relating to the constituents dimension are shown in Table A.9.

Each constituent annotation in the model representation format is stored in the corresponding `URI` sub-element of the `bioqualifier` element of the IRF (e. g. the `component OntologyEntry` relation in *mDB*). Each model annotation in the representation format is stored in the corresponding `URI` sub-element of the IRF's `modelQualifier` element (referring to the `modelOntologyEntry` relation in *mDB*). In both cases, the full URN is stored.

A list of pre-defined qualifiers is given in both the IRF and the *mDB* format. The *mDB*, in addition, defines their namespaces. The list of qualifiers currently corresponds with the list of bio-qualifiers and model-qualifiers provided by the *biomodels.net* team.

Three different types of ontology references are distinguished: Firstly, the annotation of the model with a model-qualifier is extracted from the `bqmodel:`***`any`*** annotation in the representation format, and then mapped on the `URI` element of the `modelQualifier` element in the IRF. This corresponds with the `modelOntologyURI` in *mDB*. Secondly, the annotation of a model with a bio-qualifier is extracted from the `bqbiol:`***`any`*** annotation of the `model` (`pimodel` element) in the representation format, and then mapped on the `URI` element of the `bioQualifier` element in the IRF. This also corresponds with the `modelOntologyURI` in *mDB*. Thirdly, the annotation of a constituent with a bio-qualifier is extracted from the `bqbiol:`***`any`*** annotation of any constituent in the SBML or πML model file. Those are mapped on the `URI` element of the `bioQualifier` element of any component, species, reaction, or compartment element in the IRF. This corresponds with the `bioOntologyURI` attribute in *mDB*.

Each constituent results in a single component element in the IRF (either a `species`, a `reaction`, a `compartment`, or a general `component` element), and results in a `component` entry in *mDB*. The stored information includes the components `name`, its `metaid`, its `id` (`modelComponentID` in *mDB*) in the corresponding model file. *mDB* does not have single relations for the different component types, but stores the `role` of each component. A list of pre-defined components exists.

If the IRF component is a species, then *mDB* additionally stores which reaction the species participates in. The IRF stores for each reaction a list of reactants, a list of products and a list of modifiers. All elements contained in those lists link to a species. The *mDB* then stores the reaction a species takes part in (using the `componentid`), and which role the species fulfills (`role` attribute). *mDB* also stores the location of each species in the `isLocatedIn` relation (extracted from the `compartment` element in the IRF).

User generated content dimension The concepts relating to the user generated content dimension are shown in Table A.10. A number of user-provided keywords, or tags, can be provided with each model. The `keyword` element in the IRF, and the `name` attribute of the `keyword` relation are used to store those.

The content of the `notes` elements which are allowed on any SBML and πML element in the representation format, are stored in the `notes` element of the `component` element in the IRF, and then stored in the `notes` attribute of the `component` relation (for component notes); or they are stored in the `notes` element of the `model`

RF concept		**IRF**	***mDB***
SBML	πML		
user	*user*	keyword	keyword:name
model:notes	pimodel:notes	model:modelDescription	modelAnnotation:notes
notes	notes	component:notes	component:notes

Table A.10.: Mapping of the different concepts in the user generated content dimension on the internal representation format (IRF) elements and on the *mDB* attributes, using the example of SBML and πML. The notation *xx:yy[@zz]* points to *element:attribute[@value]*; the notation *xx:yy* points to the *element:subelement* in the IRF column. The notation *relation:attribute* points to the element:attribute in the *mDB* column.

RF concept		**IRF**	***mDB***
SBML	πML		
user	*user*	experiment:experimentURL	experiment:experimentURL
user	*user*	experiment:experimentType	experiment:experimentType

Table A.11.: Mapping of the different concepts in the model-related dimension on the internal representation format (IRF) elements and on the *mDB* attributes, using the example of SBML and πML. The notation *xx:yy[@zz]* points to *element:attribute[@value]*; the notation *xx:yy* points to the *element:subelement* in the IRF column. The notation *relation:attribute* points to the element:attribute in the *mDB* column.

element in the IRF, and then stored in the `notes` attribute of the `modelAnnotation` relation (for model notes), respectively.

Model-related dimension The concepts relating to the model-related dimension are shown in Table A.11.

The representation formats SBML or πML do not link to experimental setups. However, the *mDB* allows to link experiment descriptions to a model, using the `experiment` relation. It stores an `experimentURL`, that is a link to an external experiment description, and the `experimentType`, a value from a predefined set of experiment types. The IRF allows to export that information, using the `experimentURL` and `experimentType` sub-elements of the `experiment` element. An extension of the *mDB* using a more fine-grained description of simulation experiments is suggested in Section 5.4.3. Furthermore, a similar set-up as *sDB* could be used to store information about biological experiments in a standardized way. The information on biological experiments is currently stored in the *eDB* (Section 6.6).

A.5.3. Sample search on BioModels Database demo

The following example has been published in [Henkel et al. 2010]. It illustrates the functioning of the reference implementation. We want 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`

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. A weighted list of URIs is then build and used for the feature `speciesURI` to query the MODEL INDEX. In the 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 Kanehisa and Goto [2000]. The URI (c) describes xanthine, a chemical structurally related to caffeine.

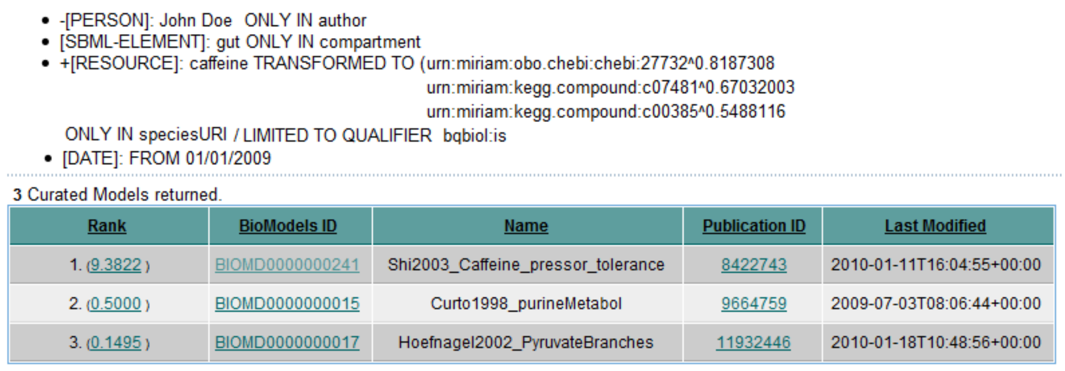
Together with the queries for `gut` in the component feature and *not* `John Doe` in the author feature, the MODEL INDEX query is internally assembled to:

```
+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 ˆ denotes the weight assigned to the sub-query results retrieved from the semantic index.

We use the Extended Boolean Model to query the index for each feature independently (compartment, speciesURI, 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.

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 7.10) 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



- -[PERSON]: John Doe ONLY IN author
- [SBML-ELEMENT]: gut ONLY IN compartment
- +[RESOURCE]: caffeine TRANSFORMED TO (urn:miriam:obo.chebi:chebi:27732^0.8187308
  urn:miriam:kegg.compound:c07481^0.67032003
  urn:miriam:kegg.compound:c00385^0.5488116

  ONLY IN speciesURI / LIMITED TO QUALIFIER bqbiol:is
- [DATE]: FROM 01/01/2009

3 Curated Models returned.

Rank	BioModels ID	Name	Publication ID	Last Modified
1. (9.3822)	BIOMD0000000241	Shi2003_Caffeine_pressor_tolerance	8422743	2010-01-11T16:04:55+00:00
2. (0.5000)	BIOMD0000000015	Curto1998_purineMetabol	9664759	2009-07-03T08:06:44+00:00
3. (0.1495)	BIOMD0000000017	Hoefnagel2002_PyruvateBranches	11932446	2010-01-18T10:48:56+00:00

Figure A.2.: The search result returned for the sample query, Figure taken from [Henkel et al. 2010]

(*must*), matches additional optional features (*should*), the scores are summed up, resulting in a higher rank. In the example, 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 in Figure A.2.

A.5.4. mDB to eDB to Mosan Object Model mapping

Table A.12 shows the mapping of *mDB* concepts and *eDB* concepts on the Mosan Object Model. Due to the similar structure of the data models inside *mDB* and Mosan (species-compartment-reaction structure), the mapping of *mDB* features onto the Mosan object structure is straight forward. All species information is provided for Mosan, together with the associated interactions. Additionally, compartmental information for each species is available. As *mDB* stores the link between a modelID in *mDB* and existing experimental data inside *eDB*, the list of existing experimental data sets can be loaded directly into Mosan and be mapped on the model structure. Another strength of Mosan is the visualization of incomplete data, which is why incomplete data sets from *eDB* can be handled easily (resulting from incomplete data import or ongoing experiments), as well as incomplete model information from *mDB* (resulting from incomplete model annotations and unstable model versions).

The conceptual idea of integrating SED-ML, *mDB*, *eDB* and Mosan has not yet been implemented. It is so far only possible to load models from *mDB* in Mosan for visualization. The remaining tasks are still under investigation (e. g. retrieving matching experiments for a loaded model, and encoding simulation results from a simulation run in SBRML).

concept	***mDB* Entity**	***eDB* Entity**	**Mosan Object Model**
model structure			
model	modelID		
	model name		model
species	speciesID		
	species name		species
	species role		*implicit*
compartment	compartmentID		
	compartmentName		
reaction	reactionID		reactionID
	participants		*implicit via IDs*
	participant role		*implicit via IDs*
annotation			
model annotation	modelOntologyEntry	experimentOntologyEntry*	
		experimentTypeAnnotation*	
entity annotation	constituentOntologyEntry	antibodyOntologyEntry*	
		entityOntologyEntry	
experiment		experimentID	experimentFile
		name	experiment name
		date	date
		user	conducter
data			
parametrisation		data values	parametrisation
(*implicit*)		result data	data
experiment description		exp. description	experiment

Table A.12.: *mDB* and *eDB* data structures mapped on the Mosan Object structure. An asterix (*) indicates conceptual features of the database that are planned to be implemented in the future versions.

A.6. Publications that have arisen from this work

The following lists the *major* reviewed publications that have arisen from this work. Please note that after my marriage in 2008 my last name changed from “Köhn” to “Waltemath”.

Journal publications

2010 Ranked Retrieval of Computational Biology Models

R. Henkel, L. Endler, A. Peters, N. Le Novère und D. Waltemath

BMC Bioinformatics 11(423)

Abstract: The study of biological systems demands computational support. If targeting a biological problem, the reuse of existing computational models can save time and effort. 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. Here we present an improved search approach for computational models of biological processes. It is based on existing retrieval and ranking methods from Information Retrieval. The approach incorporates annotations suggested by MIRIAM, and additional meta-information. It is now part of the search engine of BioModels Database, a standard repository for computational models. The introduced concept and implementation are, to our knowledge, the first application of Information Retrieval techniques on model search in Computational Systems Biology. Using the example of BioModels Database, it was shown that the approach is feasible and extends the current possibilities to search for relevant models. The advantages of our system over existing solutions are that we incorporate a rich set of meta-information, and that we provide 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.

2011 Das Sombi-Framework zum Ermitteln geeigneter Suchfunktionen für biologische Modelldatenbasen

D. Waltemath, R. Henkel, H. Meyer und A. Heuer

Datenbankspektrum

Abstract: Die Wiederverwendung von Simulationsmodellen biologischer Systeme ist mit der ansteigenden Zahl der in Modelldatenbanken gespeicherten Modelle zu einem wichtigen Forschungsproblem geworden. Ein Teilproblem ist die effiziente Suche nach relevanten Modellen in einer Datenbasis. Als Lösungsansatz wurde kürzlich die Nutzung von Information-Retrieval-Techniken fr das bewertete Finden von Modellen vorgestellt. Die im Folgenden beschriebene Software stellt Anwendungsentwicklern ein Framework zur Evaluation verschiedener Retrieval- und Rankingfunktionen unter Nutzung unterschiedlicher Datenbasen zur Verfügung. Der modulare Aufbau des Frameworks ermöglicht die Unterstützung weiterer XML-basierter Beschreibungsformate sowie das Einbinden zusätzlicher Funktionen. Voraussetzungen für die Verwendung des Frameworks sind die Kodierung der Simulationsmodelle in einem XML-basierten Standard-Repräsentationsformat sowie die Verfügbarkeit von semantischen Modellinformationen, z. B. in Form von in Ontologien kodierten Meta-Informationen. Sombi wurde als Evaluationswerkzeug für Datenbankentwickler im Bereich der Modellspeicherung in der Systembiologie entwickelt. Eine Verwendung des Frameworks auf anderen Anwendungsgebieten ist jedoch vorstellbar.

2011 Minimum Information About a Simulation Experiment (MIASE)

D. Waltemath, R. Adams, D.A. Beard, F.T. Bergmann, U.S. Bhalla, R. Britten, V. Chelliah, M.T. Cooling, J. Cooper, E. Crampin, A. Garny, S. Hoops, M. Hucka, P. Hunter, E. Klipp, C. Laibe, A. Miller, I. Moraru, D. Nickerson, P. Nielsen, M. Nikolski, S. Sahle, H. Sauro, H. Schmidt, J. Snoep, D. Tolle, O. Wolkenhauer, N. Le Novère

PLoS Computational Biology 7(4)

Abstract: Reproducibility of experiments is a basic requirement for science. Minimum Information (MI) guidelines have proved a helpful means of enabling reuse of existing work in modern biology. The MIRIAM guidelines (Minimal Information Required In the Annotation of Models) promote the exchange and reuse of biochemical computational models. However, information about a model alone is not sufficient to enable its efficient reuse in a computational setting. Advanced numerical algorithms and complex modeling workflows used in modern computational biology make reproduction of simulations difficult. It is therefore essential to define the core information necessary to perform simulations of those models. The Minimum Information About a Simulation Experiment (MIASE, glossary in Box 1) describes the minimal set of information that must be provided to make the description of a simulation experiment available to others. It includes the list of models to use and their modifications, all the simulation procedures to apply and in which order, the processing of the raw numerical results, and the description of the final output. MIASE allows for the reproduction of any simulation experiment. The provision of this information, along with a set of required models, guarantees that the simulation experiment represents the intention of the original authors. Following MIASE guidelines will thus improve the quality of scientific reporting, and will also allow collaborative, more distributed efforts in computational modeling and simulation of biological processes.

2011 Controlled vocabularies and semantics in Systems Biology

M. Courtot, N. Juty, C. Knüpfer, D. Waltemath, A. Dräger, A. Finney, M. Golebiewski, S. Hoops, S. Keating, D.B. Kell, S. Kerrien, J. Lawson, A. Lister, J. Lu, R. Machne, P. Mendes, M. Pocock, N. Rodriguez, A. Villeger, S. Wimalaratne, C. Laibe, M. Hucka und N. Le Novère

Nature Molecular Systems Biology (in the press)

Abstract: The use of computational modeling in the description and analysis of biological systems is at the heart of Systems Biology. The structure of the models, the simulation descriptions or the numerical results can be encoded in standard formats, but there is an increasing need to provide an additional semantic layer. Semantic information add meaning on components of standard descriptions to help identifying or interpreting them. Ontologies are one of the frequently used tools for such a purpose. We describe here three ontologies created specifically to address the various needs of the Systems Biology community: the Systems Biology Ontology (SBO) provides semantic information about the model components. The Kinetic Simulation Algorithm Ontology (KiSAO) supplies information about existing algorithms available for the simulation of Systems Biology models, their characterization, and inter-relationships. The Terminology for the Description of Dynamics (TEDDY) categorizes dynamical features of the simulation results and general systems behavior. The provision of semantic information extends models' longevity and facilitates their reuse. It provides useful insight into the biology of modeled processes, and may be used to make informed decisions on subsequent simulation experiments.

2011 Reproducible computational biology experiments with SED-ML – The Simulation Experiment Description Markup Language

D. Waltemath, R. Adams, F.T. Bergmann, M. Hucka, F. Kolpakov, A. Miller, I. Moraru, D. Nickerson, J.L. Snoep and N. Le Novère

submitted

Abstract: The increasing use of computational simulation experiments to inform modern biological research creates new challenges to annotate, archive, share and reproduce such experiments. The recently published Minimum Information About a Simulation Experiment (MIASE) propose a minimal set of information that should be provided to allow the reproduction of simulation experiments among users and software tools. In this article, we present an Extensible Markup Language (XML), the Simulation Experiment Description Markup Language (SED-ML). SED-ML has been developed as a community project. It is defined in a detailed technical specification and additionally provides an XML schema. The version of SED-ML described in this publication is Level 1 Version 1. It covers the description of the most frequent type of simulation experiments in the area, namely time course simulations. SED-ML encodes in a computer-readable exchange format the information required by MIASE and thereby enables reproduction of simulation experiments. SED-ML documents specify which models to use in an experiment, any modifications to apply on the models before using them, which simulation procedures to run on each model, what analysis results to output, and how the results should be presented. The description is independent of the underlying model implementation. SED-ML is a software-independent format for encoding the description of simulation experiments; it is not specific to particular simulation tools. Here, we demonstrate that with the growing software support for SED-ML we can effectively exchange executable simulation descriptions. With SED-ML, software can exchange simulation experiment descriptions, enabling the validation and reuse of simulation experiments in different tools. Authors of papers reporting simulation experiments can make their simulation protocols available for other scientists to reproduce the results. Because SED-ML is agnostic to the exact modelling language(s) used, experiments covering models from different fields of research can be accurately described and combined.

Conference proceedings

2008 A Method for Semi-Automatic Integration of Standards in Systems Biology.
D. Köhn und L. Strömbäck
Database and Expert Systems Aplications (DEXA), Lecture Notes in Computer Science

2008 SED-ML – An XML Format for the Implementation of the MIASE Guidelines
D. Köhn und N. Le Novère
Computational Methods in Systems Biology (CMSB), Lecture Notes in Bioinformatics

2009 Towards Enhanced Retrieval of Biological Models Through Annotation-based Ranking
D. Köhn, C. Maus, R. Henkel und M. Kolbe
Databases in Life Sciences (DILS), Lecture Notes in Bioinformatics

Specifications

2011 SED-ML Level 1 Version 1
D. Waltemath, F.T. Bergmann, R. Adams, N. Le Novére
Nature precedings
This specification describes the Simulation Experiment Markup Language (SED-ML) and its elements in full detail.

2011 SBML Level 3 Package Proposal: Annotation
D. Waltemath, N. Swainston, A. Lister, F.T. Bergmann, R. Henkel, S. Hoops, M. Hucka, N. Juty, S. Keating, C. Knpfer, F. Krause, C. Laibe, W. Liebermeister, C. Lloyd, G. Misirli, M. Schulz, M. Taschuk und N. Le Novère
Nature precedings
This specification describes the proposed annotation package extension to the SBML Level 3 core.

Editorship

2009 21. Workshop Grundlagen von Datenbanken
Preprints aus dem Institut für Informatik der Universität Rostock
Mathias Virgin, Andre Peters, Dagmar Köhn (Eds.)

Bibliography

L. Arévalo Rosado, A. P. Márquez, and M. S. Sánchez. An XQuery-based version extension of an XML native database. In *Proceedings of the 2009 EDBT/ICDT Workshops*, EDBT/ICDT '09, pages 99–106, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-650-2. doi: http://doi.acm.org/10.1145/1698790.1698807.

M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, May 2000. ISSN 1061-4036. doi: http://dx.doi.org/10.1038/75556.

M. Bada, R. Stevens, C. A. Goble, Y. Gil, M. Ashburner, J. A. Blake, J. M. Cherry, M. A. Harris, and S. Lewis. A short study on the success of the Gene Ontology. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(2): 235–240, February 2004. ISSN 15708268. doi: 10.1016/j.websem.2003.12.003.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval.* Addison-Wesley Harlow, England, 1999.

D. A. Beard, R. Britten, M. T. Cooling, A. Garny, M. D. Halstead, P. J. Hunter, J. Lawson, C. M. Lloyd, J. Marsh, A. Miller, D. P. Nickerson, P. M. Nielsen, T. Nomura, S. Subramanium, S. M. Wimalaratne, and T. Yu. CellML metadata standards, associated tools and repositories. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 367(1895):1845–1867, May 2009. ISSN 1364-503X. doi: 10.1098/rsta.2008.0310.

D. Beckett. RDF/XML Syntax Specification (Revised). W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

G. Bellinger, D. Castro, and A. Mills. Data, information, knowledge, and wisdom. Online publication. Accessed 16 February 2011, 2004. URL `http://www.systems-thinking.org/dikw/dikw.htm`.

P. Benjamin, M. Patki, and R. Mayer. Using ontologies for simulation modeling. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 1151–1159. IEEE press, 2006. ISBN 1-4244-0501-7. doi: http://doi.ieeecomputersociety.org/10.1109/WSC.2006.323206.

P. Benjamin, K. Akella, and A. Verma. Using ontologies for simulation integration. In *WSC '07: Proceedings of the 39th conference on Winter simulation*, pages 1081–1089, Piscataway, NJ, USA, 2007. IEEE Press. ISBN 1-4244-1306-0.

F. T. Bergmann. *An Integrative Approach to Modeling in Systems Biology.* PhD thesis, University of Washington, 2010.

F. T. Bergmann and H. M. Sauro. SBW - a modular framework for systems biology. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 1637–1645, 2006. ISBN 1-4244-0501-7.

S. Boag, D. Chamberlin, M. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML query language (Second Edition). W3C recommendation. Online publication. Accessed 16 February 2011, DEC 2010. URL `http://www.w3.org/TR/xquery/`.

S. Borneimer, M. Maurya, M. Farquhar, and S. Subramaniam. Computational modeling reveals how interplay between components of a GTPase-cycle module regulates signal transduction. *Proceedings of the National Academy of Sciences*, 101 (45):15899–15904, 2004.

B. J. Bornstein, S. M. Keating, A. Jouraku, and M. Hucka. LibSBML: an API Library for SBML. *Bioinformatics*, 24(6):880–881, March 2008. ISSN 1367-4811. doi: 10.1093/bioinformatics/btn051.

D. Brash. Reuse in Information Systems Development: Classification and Comparison. In *Proceedings of the Twelfth Australasian Conference on Information Systems*, 2001.

T. Bray, J. Paoli, E. Maler, F. Yergeau, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0 (fifth edition). W3C recommendation, W3C, Nov. 2008. http://www.w3.org/TR/2008/REC-xml-20081126/.

R. Brinkman, M. Courtot, D. Derom, J. Fostel, Y. He, P. Lord, J. Malone, H. Parkinson, B. Peters, P. Rocca-Serra, et al. Modeling biomedical experimental processes with OBI. *Journal of biomedical semantics*, 1(Suppl 1):S7, 2010.

L. D. Burgoon. The need for standards, not guidelines, in biological data reporting and sharing. *Nature Biotechnology*, 24(11):1369–1373, November 2006. ISSN 1087-0156. doi: 10.1038/nbt1106-1369.

V. Bush. As we may think. *Atlantic monthly*, 176(4):101–108, July 1945.

R. Carnap. Empiricism, semantics, and ontology. In J. Kim and E. Sosa, editors, *Metaphysics: an anthology*, chapter 2. John Wiley & Sons Ltd, 1956.

F. E. Cellier. *Continuous system modeling.* Springer-Verlag, 1 edition, December 1991. ISBN 0387975020.

V. Chelliah, L. Endler, N. Juty, C. Laibe, C. Li, N. Rodriguez, and N. Le Novère. Data Integration and Semantic Enrichment of Systems Biology Models and Simulations. In *Proceeding of the 6th International Workshop on Data Integration in the Life Sciences, DILS 2009, Manchester, UK.* Springer, July 2009.

K. C. Chen, L. Calzone, A. Csikasz-Nagy, F. R. Cross, B. Novak, and J. J. Tyson. Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell*, 15(8): 3841–3862, AUG 2004a. ISSN 1059-1524. doi: 10.1091/mbc.E03-11-0794.

M. Chen, A. Freier, J. Köhler, and A. Regg. The Biology Petri Net Markup Language. In *Lecture Notes in Informatics: Promise 2002*, 2002.

Y. Chen, S. Madria, and S. Bhowmick. DiffXML: change detection in XML data. In *Database Systems for Advanced Applications*, pages 167–196. Springer, 2004b.

J. Clark and S. DeRose. XML path language (XPath) version 1.0. W3C recommendation, W3C, Nov. 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

G. Cobena, S. Abiteboul, and A. Marian. Detecting changes in XML documents. In *Proceedings of the 18th International Conference on Data Engineering*, page 41. Published by the IEEE Computer Society, 2002. doi: http://doi.ieeecomputersociety.org/10.1109/ICDE.2002.994696.

D. L. Cook, J. L. Mejino, M. L. Neal, and J. H. Gennari. Bridging biological ontologies and biosimulation: the ontology of physics for biology. *AMIA Annual Symposium proceedings*, pages 136–140, 2008. ISSN 1942-597X.

M. T. Cooling, P. Hunter, and E. J. Crampin. Modelling biological modularity with CellML. *IET Systems Biology*, 2(2):73–79, March 2008. ISSN 17518849. doi: 10.1049/iet-syb:20070020.

M. T. Cooling, V. Rouilly, G. Misirli, J. Lawson, T. Yu, J. Hallinan, and A. Wipat. Standard virtual biological parts: a repository of modular modeling components for synthetic biology. *Bioinformatics*, 26(7):925–931, April 2010. ISSN 1367-4811. doi: 10.1093/bioinformatics/btq063.

M. Courtot, N. Juty, C. Knüpfer, D. Waltemath, A. Dräger, A. Finney, M. Golebiewski, S. Hoops, S. Keating, D. Kell, S. Kerrien, J. Lawson, A. Lister, J. Lu, R. Machne, P. Mendes, M. Pocock, N. Rodriguez, A. Villeger, S. Wimalaratne, C. Laibe, M. Hucka, and N. Le Novère. Controlled vocabularies and semantics in systems biology. *Nature Molecular Systems Biology*, 2011. in the press.

A. Cuellar, P. Nielsen, M. Halstead, D. Bullivant, D. Nickerson, W. Hedley, M. Nelson, and C. Lloyd. Cellml 1.1 specification. online, FEB 2006. available from `http://www.cellml.org/specifications/cellml_1.1`.

A. Cuellar, M. Nelson, and W. Hedley. Cellml metadata specification 1.0. Technical report, Auckland Bioengineering Institute, University of Auckland, 2009. URL `http://www.cellml.org/specifications/metadata/cellml_metadata_1.0`.

D. Cutting and J. Pedersen. Optimization for dynamic inverted index maintenance. In *Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 405–411. ACM New York, NY, USA, 1989.

J. O. Dada, I. Spasić, N. W. Paton, and P. Mendes. SBRML: a markup language for associating systems biology data with models. *Bioinformatics (Oxford, England)*, 26(7):932–938, April 2010. ISSN 1367-4811. doi: 10.1093/bioinformatics/btq069.

E. De Schutter. Why Are Computational Neuroscience and Systems Biology So Separate? *PLoS Computational Biology*, 4(5):e1000078+, May 2008. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000078.

G. Decker, H. Overdick, and M. Weske. Oryx - An Open Modeling Platform for the BPM Community. In M. Dumas, M. Reichert, and M.-C. Shan, editors, *Business Process Management*, volume 5240 of *Lecture Notes in Computer Science*, chapter 29, pages 382–385. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-85757-0. doi: 10.1007/978-3-540-85758-7_29.

K. Degtyarenko, P. de Matos, M. Ennis, J. Hastings, M. Zbinden, A. McNaught, R. Alcántara, M. Darsow, M. Guedj, and M. Ashburner. ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Research*, 36 (suppl 1):D344–D350, January 2008. ISSN 1362-4962. doi: 10.1093/nar/gkm791.

D. C. Dennett. *Freedom Evolves*, chapter 1. Penguin, new ed edition, February 2004. ISBN 0140283897.

P. J. Denning. The profession of IT: The IT schools movement. *Commun. ACM*, 44 (8):19–22, 2001. ISSN 0001-0782. doi: 10.1145/381641.381649.

F. Dyson. A meeting with Enrico Fermi. *Nature*, 427(6972):297, 2004. ISSN 0028-0836.

S. Edelstein, O. Schaad, E. Henry, D. Bertrand, and J. Changeux. A kinetic mechanism for nicotinic acetylcholine receptors based on multiple allosteric transitions. *Biological cybernetics*, 75(5):361–379, 1996.

L. Endler, N. Rodriguez, N. Juty, C. Vijayalakshmi, C. Laibe, C. Li, and N. Le Novère. Designing and encoding models for synthetic biology. *Journal of the Royal Society. Interface: suppl 4*, pages 405–417, april 2009.

D. Engmann and S. Massmann. Instance matching with coma++. In *Proceedings of the Model Management und Metadaten-Verwaltung Workshop, BTW 2007*, march 2007.

R. Ewald, C. Maus, A. Rolfs, and A. Uhrmacher. Discrete event modelling and simulation in systems biology. *Journal of Simulation*, 1:81–96, May 2007. doi: http://dx.doi.org/10.1057/palgrave.jos.4250018.

P. Eykhoff. *System Identification: Parameter and State Estimation.* John Wiley & Sons Ltd, January 1974. ISBN 0471249807.

R. Ferber. *Information Retrieval.* dpunkt.Verlag, 2003.

J. Ferrell. Q&A: Systems biology. *Journal of Biology*, 8(1):2+, January 2009. ISSN 1475-4924. doi: 10.1186/jbiol107.

A. Finkelstein, J. Hetherington, L. Li, O. Margoninski, P. Saffrey, R. Seymour, and A. Warner. Computational challenges of systems biology. *IEEE Computer*, 37(5): 26–33, 2004.

J. Fisher and T. A. Henzinger. Executable cell biology. *Nature Biotechnology*, 25 (11):1239–1249, November 2007. doi: 10.1038/nbt1356. URL `http://dx.doi.org/10.1038/nbt1356`.

P. A. Fishwick. *Simulation Model Design and Execution: Building Digital Worlds.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1995.

P. A. Fishwick and J. A. Miller. Ontologies for modeling and simulation: Issues and approaches. *Winter Simulation Conference*, 1:251–256, 2004. doi: 10.1109/WSC.2004.1371324.

D. Garfinkel. Construction of biochemical computer models. *FEBS Letters (Suppl. 1)*, pages 9–13, 1969.

J. H. Gennari, M. L. Neal, B. E. Carlson, and D. L. Cook. Integration of multi-scale biosimulation models via light-weight semantics. In *Pacific Symposium on Biocomputing*, volume 13, pages 414–425, 2008.

P. Gleeson, S. Crook, R. C. Cannon, M. L. Hines, G. O. Billings, M. Farinella, T. M. Morse, A. P. Davison, S. Ray, U. S. Bhalla, S. R. Barnes, Y. D. Dimitrova, and R. A. Silver. 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+, June 2010. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000815.

A. Goldbeter. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *Proceedings of the National Academy of Sciences of the United States of America*, 88(20):9107–9111, October 1991. ISSN 0027-8424.

G. Gottschalk, R. Kaiser, H. Malissa, E. Schwarz-Bergkampf, W. Simon, H. Spitzy, R. Werder, and H. Zettler. Systemtheorie in der analytik. *Fresenius' Journal of Analytical Chemistry*, 256(4):257–270, JAN 1971. ISSN 0016-1152. doi: 10.1007/BF00537890.

T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 2(5):199–220, June 1993.

N. Guarino. Understanding, building and using ontologies. *International Journal of Human-Computer Studies*, 46(2-3):293–310, 1997. ISSN 1071-5819. doi: 10.1006/ijhc.1996.0091.

N. Guarino. Formal ontology in information systems. In *Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy.* IOS Press, 1998. ISBN 9051993994.

A. Guyton, B. Abernathy, J. Langston, B. Kaufmann, and H. Fairchild. Relative importance of venous and arterial resistances in controlling venous return and cardiac output. *American Journal of Physiology*, 196(5):1008, 1959. ISSN 0002-9513.

R. Hälke. Annotation extraction from Computational Biology Models. Study thesis, Rostock University, 2009. Supervised by Dagmar Waltemath and Ron Henkel.

R. Hälke. Versioning Concepts and Technologies for Biochemical Simulation Models. Master's thesis, Rostock University, October 2010. Supervised by Dagmar Waltemath, Ron Henkel, Olaf Wolkenhauer, and Meike Klettke.

R. Hälke, R. Henkel, M. Klettke, D. Waltemath, and O. Wolkenhauer. The Biochemical Model Versioning System BiVeS. *in preparation*, 2011.

E. Hatcher and O. Gospodnetic. *Lucene in action*. Manning Publications, 2004.

S. Helmer, C. C. Kanne, and G. Moerkotte. XML-Datenbanksysteme und ihre Anwendung (XML Database Systems and their Application). *it–Information Technology (vormals it+ ti)/Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 45(3/2003):137–142, 2003.

R. Henkel. Determining model similarities through ranking functions using the example of biological computational models. Master's thesis, University of Rostock, September 2009. supervised by Dagmar Köhn, Andre Peters, Andreas Heuer, and Adelinde Uhrmacher.

R. Henkel, L. Endler, A. Peters, N. Le Novère, and D. Waltemath. Ranked retrieval of computational biology models. *BMC Bioinformatics*, 11(1):423+, August 2010. ISSN 1471-2105. doi: 10.1186/1471-2105-11-423.

R. Henkel, D. Waltemath, A. Peters, and A. Heuer. Retrieving and ranking computational biology models using distributed semantic knowledge. In *in preparation*, 2011.

J. Himmelspach, R. Ewald, and A. M. Uhrmacher. A flexible and scalable experimentation layer. In *Proceedings of the 2008 Winter Simulation Conference*, pages 827–835, December 2008. doi: 10.1109/WSC.2008.4736146.

M. L. Hines, T. Morse, M. Migliore, N. T. Carnevale, and G. M. Shepherd. ModelDB: A Database to Support Computational Neuroscience. *Journal of Computational Neuroscience*, 17(1):7–11–11, July 2004. ISSN 0929-5313. doi: 10.1023/B:JCNS.0000023869.22017.2e.

S. Hoops, S. Sahle, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. COPASI a COmplex PAthway SImulator. *Bioinformatics*, 22(24): 3067–3074, December 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl485.

D. Hottinger and F. Meyer. XML-Diff-Algorithmen. *Technischer Report, ETH Zürich*, july 2005.

M. Hucka, F. Bergmann, S. Hoops, S. Keating, S. Sahle, D. Wilkinson, M. Hucka, F. Bergmann, S. Hoops, S. M. Keating, S. Sahle, and D. J. Wilkinson. The systems biology markup language (sbml): Language specification for level 3 version 1 core (release 1 candidate). *Nature Precedings*, January 2010. ISSN 1756-0357. doi: 10.1038/npre.2010.4123.1. Release Candidate 1.

ISO 5807. *Information processing – Documentation symbols and conventions; program and system flowcharts.* International Organisation for Standardisation, 1985.

M. John, R. Ewald, and A. M. Uhrmacher. A spatial extension to the π calculus. In *Proceedings of the First Workshop "From Biology To Concurrency and back" (FBTC)*, volume 194 (3), pages 133–148. Elsevier Science Publishers B. V., January 2008a. doi: 10.1016/j.entcs.2007.12.010.

M. John, C. Lhoussaine, J. Niehren, and A. M. Uhrmacher. The attributed pi calculus. In *Computational Methods in Systems Biology*, pages 83–102, 2008b.

S.-H. Jung, T.-S. Jung, T.-K. Kim, K.-R. Kim, J.-S. Yoo, and W.-S. Cho. An Efficient Storage Model for the SBML Documents Using Object Databases. In M. Dalkilic, S. Kim, and J. Yang, editors, *Data mining and bioinformatics: first international workshop, VDMB 2006, Seoul, Korea, September 11, 2006: revised selected papers*, volume 4316 of *Lecture Notes in Computer Science*, chapter 9, pages 94–105. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-68970-6. doi: 10.1007/11960669_9.

N. Juty, N. Le Novère, D. Waltemath, and C. Knüpfer. Ontologies for use in systems biology: SBO, KiSAO and TEDDY. International Conference on Systems Biology (ISCB), OCT 2010. Poster presentation.

M. Kanehisa and S. Goto. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28:27.30, 2000.

V. Kashyap and A. Sheth. Semantic and schematic similarities between database objects: a context-based approach. *The VLDB Journal The International Journal on Very Large Data Bases*, 5(4):276–304, December 1996. ISSN 1066-8888. doi: 10.1007/s007780050029.

D. B. Kell and P. Mendes. The markup is the model: Reasoning about systems biology models in the Semantic Web era. *Journal of Theoretical Biology*, 252(3): 538–543, June 2008. ISSN 00225193. doi: 10.1016/j.jtbi.2007.10.023.

D. W. Kelton and R. D. Barton. Experimental design for simulation. In S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, editors, *Proceedings of the 2003 Winter Simulation Conference*, pages 60–65, 2003.

H. Kitano. *Systems Biology: Toward System-level Understanding of Biological Systems*, chapter 1. MIT Press: Cambridge, MA, 2001.

H. Kitano. Systems biology: A brief overview. *Science*, 295(5560):1662–1664, March 2002a. ISSN 1095-9203. doi: 10.1126/science.1069492.

H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, November 2002b. ISSN 0028-0836. doi: 10.1038/nature01254.

A. Klaus and W. Birchmeier. Wnt signalling and its impact on development and cancer. *Nature Reviews Cancer*, 8(5):387–398, May 2008. ISSN 1474-175X. doi: 10.1038/nrc2389.

M. Klettke and H. Meyer. *XML und Datenbanken*, volume 1. dunkt.verlag, 2003a.

M. Klettke and H. Meyer. Speicherung von XML-Dokumenten - eine Klassifikation. *Datenbank-Spektrum*, 5:40–50, 2003b.

E. Klipp and W. Liebermeister. Mathematical modeling of intracellular signaling pathways. *BMC Neuroscience*, 7(Suppl 1):S10+, 2006. ISSN 1471-2202. doi: 10.1186/1471-2202-7-S1-S10.

E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach. *Systems Biology in Practice. Concepts, Implementation and Appplication.* WILEY-VCH Verlag GmbH & Co. KGaA, 3rd reprint edition, 2007a.

E. Klipp, W. Liebermeister, A. Helbig, A. Kowald, and J. Schaber. Systems biology standards - the community speaks. *Nature Biotechnology*, 25(4):390–391, April 2007b. ISSN 1087-0156. doi: 10.1038/nbt0407-390.

E. Klipp, W. Liebermeister, C. Wierling, A. Kowald, H. Lehrach, and R. Herwig. *Systems Biology: A Textbook.* Wiley-VCH, 2009.

C. Knüpfer, C. Beckstein, and P. Dittrich. Towards a semantic description of biomodels: Meaning facets – a case study. In *Proceedings of the Second International Symposium on Semantic Mining in Biomedicine (SMBM 2006), Jena, April 9th-12th. CEUR-WS*, 2006.

D. Köhn. Enhancing model reuse in systems biology using an integrative storage and retrieval system. In *Proceedings of the 2009 Joint EDBT/ICDT Ph.D. workshop*. ACM, March 2009.

D. Köhn and M. John. Making the π calculus exchangable – the Pi Markup Language (PiML), 2007. Conference on Computational Methods in Systems Biology, Edinburgh (poster presentation).

D. Köhn and N. Le Novère. SED-ML – An XML Format for the Implementation of the MIASE Guidelines. In M. Heiner and A. M. Uhrmacher, editors, *Computational Methods in Systems Biology*, volume 5307/2008 of *Lecture Notes in Systems Biology*, pages 176–190. Springer-Verlag Berlin Heidelberg, October 2008. doi: 10.1007/978-3-540-88562-7_15.

D. Köhn and L. Strömbäck. A method for semi-automatic integration of standards in systems biology. In *19th International Conference on Database and Expert Systems Aplications (DEXA)*, volume 5181 (0745) of *Lecture Notes in Computer Science*, pages 745–752. Springer, 2008.

D. Köhn, C. Maus, R. Henkel, and M. Kolbe. Towards enhanced retrieval of biological models through annotation-based ranking. In N. W. Paton, P. Missier, and C. Hedeler, editors, *Data Integration in the Life Sciences, 6th International Workshop, DILS*, volume 5647 of *Lecture Notes in Bioinformatics*, pages 204–219, Manchester, UK, 2009. Springer.

M. Kolbe. MIRIAM based storage and query of biological simulation models. Master's thesis, University of Rostock, June 2009. Supervised by Dagmar Köhn, Carsten Maus, Andreas Heuer and Adelinde Uhrmacher.

F. Krause, J. Uhlendorf, T. Lubitz, M. Schulz, E. Klipp, and W. Liebermeister. Annotation and merging of SBML models with semanticSBML. *Bioinformatics*, pages btp642+, November 2009. doi: 10.1093/bioinformatics/btp642.

R. Kühn. Kopplung von XML-Schema Komponenten am Beispiel von πML. Study thesis, Rostock Unversity, 2008. Supervised by Dagmar Köhn, Mathias John and Andreas Heuer.

D. Kuropka. *Modelle zur Repräsentation natürlichsprachlicher Dokumente. Ontologie-basiertes Information-Filtering und -Retrieval mit relationalen Datenbanken.* Logos Verlag, 2004. ISBN 978-3-8325-0514-1.

C. Laibe and N. Le Novère. MIRIAM Resources: tools to generate and resolve robust cross-references in Systems Biology. *BMC Systems Biology*, 1(1), December 2007. doi: 10.1186/1752-0509-1-58.

P. Lambrix. Towards a Semantic Web for Bioinformatics using Ontology-based Annotation. In *Proceedings oof the 14th IEEE International Workshop on Enabling Technologies: Infrastructure of Collaborative Enterprises*, Linköping, Sweden, January 2005. IEEE.

P. Lambrix, L. Strömbäck, and H. Tan. *Semantic Techniques for the Web*, chapter 8 - Information Integration in Bioinformatics with Ontologies and Standards, pages 343–376. Springer-Verlag Berlin Heidelberg, LNCS 5500 edition, 2009.

N. Le Novère. Model storage, exchange and integration. *BMC Neuroscience*, 7 (Suppl 1), 2006. doi: 10.1186/1471-2202-7-S1-S11.

N. Le Novère. Principled annotation of quantitative models in systems biology. Presentation slides, 2008. Genomes to Systems, Manchester, UK. Available at `http://www.ebi.ac.uk/~lenov/LECTURES/G2S-LeNovere.pdf`.

N. Le Novère. MIBBI, MIASE and all that. Presentation slides, April 2009. URL `http://www.cellml.org/community/events/workshop/2009/presentations/nicolas_mibbi.pdf`. The combined CellML-SBGN-SBO-BioPAX-MIASE 2009 workshop. Auckland, New Zealand.

N. Le Novère and A. Finney. A simple scheme for annotating SBML with references to controlled vocabularies and database entries. December 2005. URL `http://www.ebi.ac.uk/compneur-srv/sbml/proposals/AnnotationURI.pdf`.

N. Le Novère, A. Finney, M. Hucka, U. S. Bhalla, F. Campagne, J. Collado-Vides, E. J. Crampin, M. Halstead, E. Klipp, P. Mendes, P. Nielsen, H. Sauro, B. Shapiro, J. L. Snoep, H. D. Spence, and B. L. Wanner. Minimum Information Requested In the Annotation of biochemical Models (MIRIAM). *Nature Biotechnology*, 23 (12):1509–1515, December 2005. ISSN 1087-0156. doi: 10.1038/nbt1156.

N. Le Novère, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, J. L. Snoep, and M. Hucka. Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34(Database issue), January 2006. ISSN 1362-4962.

N. Le Novère, M. Courtot, and C. Laibe. Adding semantics in kinetics models of biochemical pathways. In *Proceedings of the 2nd International Symposium on Experimental Standard Conditions of Enzyme Characterizations (ESEC 2006), Beilstein Institute, Frankfurt am Main, Germany*, 2007.

N. Le Novère, M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M. I. Aladjem, S. M. Wimalaratne, F. T. Bergman, R. Gauges, P. Ghazal, H. Kawaji, L. Li, Y. Matsuoka, A. Villeger, S. E. Boyd, L. Calzone, M. Courtot, U. Dogrusoz, T. C. Freeman, A. Funahashi, S. Ghosh, A. Jouraku, S. Kim, F. Kolpakov, A. Luna, S. Sahle, E. Schmidt, S. Watterson, G. Wu, I. Goryanin, D. B. Kell, C. Sander, H. Sauro, J. L. Snoep, K. Kohn, and H. Kitano. The Systems Biology Graphical Notation. *Nature Biotechnology*, 27(8):735–741, August 2009. ISSN 1087-0156. doi: 10.1038/nbt.1558.

C. Lehner. Beitrag zu einer holistischen Theorie für die Informationswissenschaften. *Fortschritte der Wissensorganisation ISKO/Hamburg*, 1999.

Simplex3: EDL Referenzbuch. Lehrstuhl für Operations Research und Systemtheorie der Universität Passau, 2008. URL `http://wwwisg.cs.ovgu.de/sim/its/ws0405/simplex/edl.pdf`.

J. Leloup and A. Goldbeter. Chaos and birhythmicity in a model for circadian oscillations of the PER and TIM proteins in Drosophila. *Journal of theoretical biology*, 198(3):445–459, 1999.

J. Leloup, D. Gonze, and A. Goldbeter. Limit cycle models for circadian rhythms based on transcriptional regulation in Drosophila and Neurospora. *Journal of Biological Rhythms*, 14(6):433, 1999.

C. Li, M. Donizelli, N. Rodriguez, H. Dharuri, L. Endler, V. Chelliah, L. Li, E. He, A. Henry, M. Stefan, J. Snoep, M. Hucka, N. Le Novère, and C. Laibe. Biomodels database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4(1):92+, June 2010. ISSN 1752-0509. doi: 10.1186/1752-0509-4-92.

W. Liebermeister. Validity and combination of biochemical models. In *Proceedings of 3rd International ESCEC Workshop on Experimental Standard Conditions on Enzyme Characterizations*, september 2007.

D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, July 1998.

A. L. Lister, M. Pocock, M. Taschuk, and A. Wipat. Saint: a lightweight integration environment for model annotation. *Bioinformatics*, 25(22):3026–3027, November 2009. ISSN 1367-4811. doi: 10.1093/bioinformatics/btp523.

C. M. Lloyd, M. D. B. Halstead, and P. F. Nielsen. CellML: its future, present and past. *Progress in Biophysics and Molecular Biology*, 85:433–450, february 2004.

C. M. Lloyd, J. R. Lawson, P. J. Hunter, and P. F. Nielsen. The CellML Model Repository. *Bioinformatics*, 24(18):2122–2123, September 2008. ISSN 1460-2059. doi: 10.1093/bioinformatics/btn390.

R. Magjarevic, T. Yu, J. R. Lawson, and R. D. Britten. A Distributed Revision Control System for Collaborative Development of Quantitative Biological Models. In C. T. Lim and J. Goh Cho Hong, editors, *13th International Conference on Biomedical Engineering*, volume 23 of *IFMBE Proceedings*, chapter 473, pages 1908–1911. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-540-92841-6_473.

L. Marenco, N. Tosches, C. Crasto, G. Shepherd, P. L. Miller, and P. M. Nadkarni. Achieving Evolvable Web-Database Bioscience Applications Using the EAV/CR Framework: Recent Advances. *Journal of the American Medical Informatics Association*, 10(5):444–453, SEP 2003. doi: 10.1197/jamia.M1303.

W. Materi and D. S. Wishart. Computational systems biology in drug discovery and development: methods and applications. *Drug Discovery Today*, 12(7-8):295–303, April 2007. ISSN 13596446. doi: 10.1016/j.drudis.2007.02.013.

D. L. McGuinness and F. van Harmelen. OWL web ontology language overview. W3C recommendation, W3C, Feb. 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

A. D. McNaught and A. Wilkinson. *IUPAC Compendium of Chemical Terminology*. WileyBlackwell, 2nd revised edition edition, August 1997. ISBN 0865426848.

A. Miller. Simulation Metadata Specification. Online publication. Accessed 16 February 2011, April 2009. URL `http://www.cellml.org/specifications/metadata`.

E. Miller and F. Manola. RDF primer. W3C recommendation, W3C, Feb. 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

J. A. Miller, G. T. Baramidze, A. P. Sheth, and P. A. Fishwick. Investigating ontologies for simulation modeling. In *Proceedings of the 37th Annual Simulation Symposium (ANSS04)*. IEEE, 2004.

R. J. Miller, M. A. Hernandez, L. M. Haas, L. Yan, H. C. T. Ho, R. Fagin, and L. Popa. The Clio project: managing heterogeneity. *SIGMOD Rec.*, 30(1):78–83, March 2001. ISSN 0163-5808. doi: 10.1145/373626.373713.

R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, June 1999. ISBN 0521658691.

M. Minsky. Models, minds, machines. In *Proceedings of the IFIP Congress*, pages 45–49, 1965.

J. Montani, T. Adair, R. Summers, T. Coleman, and A. Guyton. A simulation support system for solving large physiological models on microcomputers. *International journal of bio-medical computing*, 24(1):41, 1989. ISSN 0020-7101.

I. I. Moraru, J. C. Schaff, B. M. Slepchenko, M. L. Blinov, F. Morgan, A. Lakshminarayana, F. Gao, Y. Li, and L. M. Loew. Virtual Cell modelling and simulation software environment. *IET Systems Biology*, 2(5):352–362, September 2008. ISSN 17518849. doi: 10.1049/iet-syb:20080102.

Nature. In Pursuit of Systems. *Nature*, 435(1), 2005. Editorial.

D. P. Nickerson and M. L. Buist. A physiome standards-based model publication paradigm. *Physical and Engineering Sciences*, 367(1895):1823–1844, May 2009. doi: 10.1098/rsta.2008.0296.

D. P. Nickerson, A. Corrias, and M. L. Buist. Reference descriptions of cellular electrophysiology models. *Bioinformatics*, 24(8):1112–1114, March 2008. doi: 10.1093/bioinformatics/btn080.

E. Nordlie, M.-O. Gewaltig, and H. E. Plesser. Towards Reproducible Descriptions of Neuronal Network Models. *PLoS Comput Biol*, 5(8):e1000456+, August 2009. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000456.

B. Novak and J. J. Tyson. Numerical analysis of a comprehensive model of M-phase control in Xenopus oocyte extracts and intact embryos. *Journal of Cell Science*, 106(4):1153–1168, December 1993. URL `http://jcs.biologists.org/content/106/4/1153.abstract`.

R. Nusse. Wnt signaling and stem cell control. *Cell research*, 18(5):523–527, May 2008. ISSN 1748-7838. doi: 10.1038/cr.2008.47.

Y. Oertel. ExML – ein Austauschformat für valide Experimentbeschreibungen zum Austausch zwischen verschiedenen Simulationssystemen. Master's thesis, Rostock University, Fakultät für Informatik und Elektrotechnik, April 2009. Supervised by Dagmar Köhn, Jan Himmelspach, Adelinde M Uhrmacher and Olaf Wolkenhauer.

B. G. Olivier and J. L. Snoep. Web-based kinetic modelling using JWS online. *Bioinformatics (Oxford, England)*, 20(13):2143–2144, September 2004. ISSN 1367-4803. doi: 10.1093/bioinformatics/bth200.

C. Osgood, G. Suci, and P. Tannenbaum. *The measurement of meaning.* University of Illinois Press, 1971.

H. Ou. Speicherung Komplexer XML-Modelle am Beispiel von SBML. Study thesis, Rostock University, April 2009. Supervised by Dagmar Köhn.

C. Overstreet, R. Nance, and O. Balci. Issues in enhancing model reuse. In *International Conference on Grand Challenges for Modeling and Simulation*, pages 27–31, 2002.

G. Pastori, V. Simons, and M. van Bogaert. Systems biology in the european research area. ERASysBio Partners, strategy paper, March 2008.

K. Pawlikowski, H. D. J. Jeong, and J. R. Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications Magazine*, 40:132–139, 2002.

H. A. Piwowar, R. S. Day, and D. B. Fridsma. Sharing Detailed Research Data Is Associated with Increased Citation Rate. *PLoS ONE*, 2(3):e308+, 2007. ISSN 1932-6203. doi: 10.1371/journal.pone.0000308.

N. Poppelier, R. Miner, P. Ion, and D. Carlisle. Mathematical markup language (MathML) version 2.0 (second edition). W3C recommendation, W3C, Oct. 2003. http://www.w3.org/TR/2003/REC-MathML2-20031021/.

M. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

C. Priami. Stochastic π-calculus. *The Computer Journal*, 38(7):578, 1995. ISSN 0010-4620.

C. Priami and P. Quaglia. Modelling the dynamics of biosystems. *Briefings in Bioinformatics*, 5(3):259–269, 2004. doi: 10.1093/bib/5.3.259. URL `http://bib.oxfordjournals.org/content/5/3/259.abstract`.

J. Quackenbush. Standardizing the standards. *Molecular systems biology*, 2:1–3, February 2006. ISSN 1744-4292. doi: 10.1038/msb4100052.

R. Reese and D. L. Wyatt. Software reuse and simulation. In *WSC '87: Proceedings of the 19th conference on Winter simulation*, pages 185–192, New York, NY, USA, 1987. ACM. ISBN 0-911801-32-4. doi: 10.1145/318371.318404.

S. Robinson, R. Nance, R. Paul, M. Pidd, and S. Taylor. Simulation model reuse: definitions, benefits and obstacles. *Simulation Modelling Practice and Theory*, 12 (7-8):479–494, November 2004. ISSN 1569190X. doi: 10.1016/j.simpat.2003.11.006.

M. Röhl. *Definition und Realisierung einer Plattform zur modellbasierten Komposition von Simulationsmodellen.* PhD thesis, Rostock University, 2008.

M. Röhl and A. M. Uhrmacher. Flexible integration of XML into modeling and simulation systems. In *Proceedings of the 38th Winter simulation Conference*, pages 1813–1820, December 2005. doi: 10.1145/1013329.1013349.

S. Rönnau, J. Scheffczyk, and U. M. Borghoff. Towards xml version control of office documents. In *Proceedings of the 2005 ACM symposium on Document engineering*, DocEng '05, pages 10–19, New York, NY, USA, 2005. ACM. ISBN 1-59593-240-2. doi: http://doi.acm.org/10.1145/1096601.1096606.

C. Rosse and J. Mejino. A reference ontology for biomedical informatics: the Foundational Model of Anatomy. *Journal of biomedical informatics*, 36(6):478–500, 2003. ISSN 1532-0464.

J. Rowley and R. Hartley. *Organizing knowledge: an introduction to managing access to information*, chapter Knowledge, information, and their organisation. Ashgate Publishng Limited, 2008.

P. Saffrey and R. 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*, 18:613–620, November 1975. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/361219.361220.

D. Sangiorgi and D. Walker. *Pi-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001. ISBN 0521781779.

H. M. Sauro, A. M. Uhrmacher, D. Harel, M. Hucka, M. Kwiatkowska, P. Mendes, C. A. Shaffer, L. Strömbäck, and J. J. Tyson. Challenges for modeling and simulation methods in systems biology. In L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, editors, *Proceedings of the 2006 Winter Simulation Conference*, pages 1720–1730, December 2006.

I. Schmitt. *Ähnlichkeitssuche in Multimedia-Datenbanken*. Oldenbourg Wissensch.Vlg, 2005.

M. Schulz, F. Krause, N. Le Novère, E. Klipp, and W. Liebermeister. Retrieval, alignment, and clustering of computational models based on semantic annotations. *submitted*, 2010.

D. J. Sherman. Minimum information requirements: Neither bandits in the attic nor bats in the belfry. *New Biotechnology*, 25(4):173–174, April 2009.

A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22:183–236, 1990.

G. A. Silver, O. A. H. Hassan, and J. A. Miller. From domain ontologies to modeling ontologies to executable simulation models. In *WSC '07: Proceedings of the 39th conference on Winter simulation*, pages 1108–1117, Piscataway, NJ, USA, 2007. IEEE Press. ISBN 1-4244-1306-0.

B. Smith. Ontology definitions. Online publication. Accessed 25 September 2010., 2010. URL `http://www.ehealthserver.com/ontology/index.php?option=com_content&task=view&id=5&Itemid=6`.

B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. H. Scheuermann, N. Shah, P. L. Whetzel, and S. Lewis. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, November 2007. ISSN 1087-0156. doi: 10.1038/nbt1346.

J. L. Snoep and B. G. Olivier. JWS Online Cellular Systems Modelling and Microbiology. *Microbiology*, 149(11):3045–3047, November 2003. doi: http://dx.doi.org/10.1099/mic.0.C0124-0.

L. N. Soldatova and R. D. King. Are the current ontologies in biology good ontologies? *Nature Biotechnology*, 23(9):1095–1999, 2005.

M. Spiegel, P. F. Reynolds, and D. C. J. Brogan. A case study of model context for simulation composability and reusability. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, editors, *Proceedings of the 2005 Winter Simulation Conference*, 2005.

R. Stevens, A. Rector, and D. Hull. What is an ontology? *Ontogenesis – Online Journal. Accessed 2 April 2010*, January 2010. URL `http://ontogenesis.knowledgeblog.org/2010/01/22/what/`.

L. Strömbäck, D. Hall, and P. Lambrix. A review of standards for data exchange within systems biology. *Proteomics*, 7(6):857–867, March 2007. ISSN 1615-9853. doi: 10.1002/pmic.200600438.

N. Swainston. SBML Browse. Poster presentation, SBML Hackathon 2010, March 2010. URL `http://www.mcisb.org/SBMLBrowse/`.

N. Swainston and P. Mendes. libAnnotationSBML: a library for exploiting SBML annotations. *Bioinformatics*, 25(17):2292–2293, September 2009. ISSN 1460-2059. doi: 10.1093/bioinformatics/btp392.

C. Taylor, D. Field, S. Sansone, J. Aerts, R. Apweiler, M. Ashburner, C. Ball, P. Binz, M. Bogue, T. Booth, et al. Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the MIBBI project. *Nature Biotechnology*, 26(8):889–896, 2008.

J. J. Tyson. Modeling the cell division cycle: cdc2 and cyclin interactions. *Proceedings of the National Academy of Sciences of the United States of America*, 88(16): 7328–7332, August 1991. doi: 10.1073/pnas.88.16.7328.

H. Ueda, M. Hagiwara, and H. Kitano. Robust oscillations within the interlocked feedback model of Drosophila circadian rhythm. *Journal of Theoretical Biology*, 210(4):401–406, 2001.

A. M. Uhrmacher, M. John, O. Mazemondet, A. Unger, T. Rharass, B. Bader, and A. Rolfs. Computer Science Meets Cell Biology – GRK dIEM oSiRiS. Technical report, Rostock University, 2009. Rostocker Informatikberichte.

A. Unger. *Visual Support for the Modeling and Simulation of Cell Biological Processes*. PhD thesis, University of Rostock, 2010.

A. Unger and H. Schumann. Visual support for the understanding of simulation processes. *Visualization Symposium, IEEE Pacific*, 0:57–64, 2009. doi: 10.1109/PACIFICVIS.2009.4906838.

A. Unger, S. Biermann, M. John, A. Uhrmacher, and H. Schumann. Visual support for modeling and simulation of cell biological systems. Poster, Washington, D.C., USA, Winter Simulation Conference, December 2007.

C. J. Van Rijsbergen. *Information Retrieval.* London: Butterworths, 2nd edition, 1979. URL `http://www.dcs.gla.ac.uk/Keith/Preface.html`.

K. Verspoor, D. Dvorkin, B. B. Cohen, and L. Hunter. Ontology quality assurance through analysis of term transformations. *Bioinformatics (Oxford, England)*, 25 (12):i77–84, June 2009. ISSN 1367-4811. doi: 10.1093/bioinformatics/btp195.

A. C. Villéger, S. R. Pettifer, and D. B. Kell. Arcadia: a visualization tool for metabolic pathways. *Bioinformatics (Oxford, England)*, 26(11):1470–1471, June 2010. ISSN 1367-4811. doi: 10.1093/bioinformatics/btq154.

W3C. XML Schema Structures, October 2004. URL `http://www.w3.org/TR/xmlschema-1/`. `http://www.w3.org/TR/xmlschema-1/`.

D. Waltemath, R. Adams, D. A. Beard, F. T. Bergmann, U. S. Bhalla, R. Britten, V. Chelliah, M. T. Cooling, J. Cooper, E. Crampin, A. Garny, S. Hoops, M. Hucka, P. Hunter, E. Klipp, C. Laibe, A. Miller, I. Moraru, D. Nickerson, P. Nielsen, M. Nikolski, S. Sahle, H. Sauro, H. Schmidt, J. L. Snoep, D. Tolle, O. Wolkenhauer, and N. Le Novère. Minimum Information About a Simulation Experiment (MIASE). *PLoS Computational Biology (in the press)*, 2011a.

D. Waltemath, F. Bergmann, R. Adams, and N. Le Novère. *Simulation Experiment Description Markup Language (SED-ML): Level 1 Version 1*, 2011b. URL `http://dx.doi.org/10.1038/npre.2011.5846.1`. Available from Nature Precedings.

D. Waltemath, R. Henkel, H. Meyer, and A. Heuer. Das Sombi-Framework zum Ermitteln geeigneter Suchfunktionen für biologische Modelldatenbasen. *Datenbank-Spektrum*, pages 1–10, 2011c. ISSN 1618-2162. doi: http://dx.doi.org/10.1007/s13222-011-0050-x.

D. Waltemath, N. Swainston, A. Lister, F. Bergmann, R. Henkel, S. Hoops, M. Hucka, N. Juty, S. Keating, C. Knüpfer, F. Krause, C. Laibe, W. Liebermeister, C. Lloyd, G. Misirli, M. Schulz, M. Taschuk, and N. Le Novère. SBML Level 3 Package Proposal: Annotation. Nature Precedings, 2011d.

M. Weitzel. Adapting Knowledge Management Solutions to Computational Systems Biology: Introducing a Systems Biology Portal Server. Study thesis, Rostock University, 2011 (to appear). Supervised by Ron Henkel, Holger Meyer and Dagmar Waltemath.

S. White. Introduction to BPMN. *IBM Cooperation*, pages 2008–029, 2004.

S. M. Wimalaratne, M. D. B. Halstead, C. M. Lloyd, M. T. Cooling, E. J. Crampin, and P. F. Nielsen. A method for visualizing CellML models. *Bioinformatics*, 25 (22):3012–3019, November 2009a. doi: 10.1093/bioinformatics/btp495.

S. M. Wimalaratne, M. D. B. Halstead, C. M. Lloyd, E. J. Crampin, and P. F. Nielsen. Biophysical annotation and representation of CellML models. *Bioinformatics*, 25(17):2263–2270, September 2009b. ISSN 1460-2059. doi: 10.1093/bioinformatics/btp391.

D. Wolf and A. P. Arkin. Motifs, modules and games in bacteria. *Current Opinion in Microbiology*, 6(2):125–134, April 2003. ISSN 13695274. doi: 10.1016/S1369-5274(03)00033-X.

B. Zhang, J. Shen, Q. Xiang, and Y. Wang. Compositemap: A novel framework for music similarity measure. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 403–410. ACM, 2009.

Theses

1. The reuse of models supports research in systems biology, but it is hindered by the difficulty in retrieving relevant models. This thesis argues that the development of enhanced search techniques would improve model reuse.

2. Many tasks related to bio-models cannot solely rely upon model encoding, and must also incorporate processable meta-information. A rich and unambiguous annotation scheme is desirable. This work implements such a scheme for the model representation format SBML.

3. Current model repositories are restricted to one particular model representation format. This situation necessitates the repeated development of model management capabilities for each repository, in particular to realize model versioning, display, and search. This work shows how a format-independent, generic model database allows for integrative model maintenance.

4. Current search systems are format-specific, and the incorporation of model meta-information is limited. The search system developed in this thesis abstracts from the model encoding and evaluates available meta-information. It enables search across different model representation formats, and leads to richer and more complete search results.

5. The results of conventional model search are presented as simple sets of model IDs. A ranked retrieval system, as developed in this work, supports users in finding the best matching model for their question.

6. Models evolve as new details are unraveled, while previous versions of a model must retain addressable. This work demonstrates how a versioning system for bio-models tracks both, changes in the model and in its annotations.

7. Reproducibility of results is a basic requirement in systems biology, not only for experiments in the wet lab, but also for models and simulation. Finding a model in a database is not sufficient to fully comprehend the meaning of a model. Hence the simulation experiments performed upon it must also be made available. In this work, a standard format for the encoding of simulation experiments is developed.

In desperation I asked Fermi whether he was not impressed by the agreement between our calculated numbers and his measured numbers. He replied, How many arbitrary parameters did you use for your calculations? I thought for a moment about our cut-off procedures and said, Four. He said, I remember my friend Johnny von Neumann used to say, with four parameters I can fit an elephant, and with five I can make him wiggle his trunk. With that, the conversation was over.

(F. Dyson, [Dyson 2004])

Selbstständigkeitserklärung

Ich erkläre, dass ich die eingereichte Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Dagmar Waltemath

Rostock, 23. März 2011