

Improving Reproducibility and Reuse of Modelling Results in the Life Sciences

Martin Scharm



SYSTEMS BIOLOGY
BIOINFORMATICS
ROSTOCK

Improving Reproducibility and Reuse of Modelling Results in the Life Sciences

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

Promotionsgebiet Systembiologie und Bioinformatik

Fakultät für Informatik und Elektrotechnik
Universität Rostock

Universität
Rostock



Traditio et Innovatio

vorgelegt von
Martin Scharm, geboren am 27. Oktober 1985 in Güstrow
Rostock, 5. Oktober 2018

Gutachter:	Prof. Olaf Wolkenhauer	Universität Rostock
	Prof. Ina Koch	Goethe-Universität Frankfurt am Main
	Prof. Clemens H. Cap	Universität Rostock
Verteidigung:	30. August 2018	



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Compiled Friday 5th October, 2018 at 13:54. This is pdfTeX, Version 3.14159265-2.6-1.40.19 (TeX Live 2019/dev/Debian) kpathsea version 6.3.1/dev

To my grandfather Herwig Schmidt,
who said it would be a good thing to do.

ACKNOWLEDGEMENTS

I want to express special gratitude to my advisor Prof. Olaf Wokenhauer, henceforth referred to as the *Kaleun*. I am truly impressed, how he stood my habits of shamelessly taking liberties. His (often subtle) guidance regularly let me see opportunities or got me back on the straight and narrow – whatever was necessary. As head of the department of Systems Biology and Bioinformatics (SBI), the Kaleun established an interdisciplinary and intercultural environment full of ideas, perspectives, and opinions. I am glad to experience this atmosphere.

Many colleagues at the SBI became close friends. First of all, Dagmar Waltemath, you got me to Rostock and together we are riding a successful wave. A fruitful collaboration turned into a prosperous friendship, and ultimately inspired this thesis' layout. Furthermore, I would like to thank all my colleagues, who were always up for discussing work, life, and ... everything! I remember numerous happy or ruminating faces of Ali Salehzadeh-Yazdi, Holger Hennig, Sherry Freiesleben, Ulf Liebal, Shailendra Gupta, and Vasundra Touré.

I need to highlight the *Hochleistungslabor*, and its team (and religion). The happenings with Tom Theile, Markus Wolfien, Martin Peters, and Tom Gebhardt committed to my memory. Even though it probably seemed foolish from the outside, we learnt a lot while playing with the environment and questioning established (infra-)structures just like children. Our office and the competent non-professionalism cleared space to just *do*, which engendered projects that (i) usually totally failed, (ii) sometimes made others rethink, or in one case (iii) reached millions of people! Over the time, the *Hochleistungslabor* (and its religion) gained a few disciples. The debates, drinks, and good music with Henning Hinze, Florian Wendland, and Andrea Bagnacani often made my week.

I always enjoyed having students and pupils around me, who typically raised untainted questions worth a million conference talks. Mariam Nassar, Fabienne Lambusch, Vivek Garg, Srijana Kayastha, and Jenny Fabian have a substantial stake in this work. Together, we fought

for interoperability and reproducibility. Many pupil interns made me experiment with the uncommon and actually had an impact on tools and ideas presented in this thesis.

I also want to thank all collaborators for their bold scientific input. I am especially grateful to the Science 2020 team in Oxford and the Information Management Group in Manchester, which both hosted me for quite some time. In particular, Jonathan Cooper, Stian Soiland-Reyes, Natalie Stanford, Matthew Gamble, Gary Mirams, and Carole Goble substantially influenced me and my ideas. Moreover, would like to thank the whole COMBINE community. Many discussions and meetings significantly increased the robustness of my methods and implementations. Above all, Pedro Mendes, Tommy Yu, Frank Bergmann, Olga Krebs, and Falk Schreiber helped a lot to disseminate my tools.

Last but not least, I want to say a big thank-you to family and friends, who assisted in focussing with regular reminders of the unfinished project (with a surprisingly dynamic range of empathy). Obviously, my private life suffered to make me achieve this goal. After all, I forked a human being while writing up! However, I promise to improve.

All these fragrant memories, when writing this section, make me feel that I could dedicate a whole chapter on acknowledgements of people, stories, songs, bands, breweries, pubs, etc. that altogether made my time as a PhD student a great success. Unfortunately, I am running out of

ABSTRACT

Modelling and simulation is a standard tool for research in the life sciences. Moreover, distributed and collaborative modelling has become increasingly prominent. The growing impact of simulation studies is reflected by a rapidly increasing number of computational models in open model repositories. Research results in the life sciences are typically complex and include a variety of heterogeneous data. These developments, however, entail a number of computational challenges to facilitate reproducibility and reuse of modelling results. Above all, it is difficult (i) to manage simulation studies, (ii) to ensure model exchangeability, stability and validity, and (iii) to foster communication between project partners.

In this thesis, I present techniques to improve the reproducibility and reuse of modelling results in the life sciences. First, I introduce a method to characterise differences in computational models. At the heart of my method, an algorithm is able to identify differences between models in standard format and an ontology can be used to semantically describe identified changes. I show how my method supports researchers in grasping a model's evolution and improves collaborative modelling. Second, I developed multiple approaches to obtain shareable and reproducible research results. A novel container format eases the handling of simulation studies. I present various tools to find, explore, create, and share research results. In addition, I describe my fully featured demo study and show how it can be reproduced.

The solutions presented in this thesis do not just exist in theory. Instead, I materialised most of my ideas into code and a number of reusable tools emerged from this work. I show how my tools have already been successfully integrated in several third-party applications, demonstrating the impact of my ideas and implementations in the systems biology community. Altogether, my methods and tools foster exchange and reuse of modelling results.

ZUSAMMENFASSUNG

Die Modellierung und Simulation biologischer Systeme mit Hilfe des Computers gehört inzwischen zum Standardrepertoire in den Lebenswissenschaften. Ein Modell wird dabei jedoch nicht immer von einem einzelnen Wissenschaftler entwickelt, sondern entsteht oft in Zusammenarbeit von Forschern aus einem interdisziplinären Umfeld und basiert häufig auf existierenden Modellen. Diese verteilte Entwicklung von verhältnismäßig komplexen Simulationsstudien birgt eine große Zahl an informationstechnischen Herausforderungen: (i) Die Modelle müssen verwaltet werden; (ii) die Reproduzierbarkeit, Stabilität und Gültigkeit von Simulationsstudien und daraus resultierenden Ergebnissen muss sichergestellt werden; und (iii) die Kommunikation zwischen Kollaborationspartnern muss verbessert werden.

In dieser Arbeit stelle ich verschiedene Techniken vor, um die Reproduzierbarkeit und Wiederverwendbarkeit von Modellierungsergebnissen in den Lebenswissenschaften zu verbessern. Ich führe zunächst eine Methode ein, mit der Unterschiede zwischen Modellversionen erkannt und charakterisiert werden können. Das daraus resultierende, transparente Verständnis der Evolution eines Modells hilft Änderungen, die beispielsweise von Kollegen und Kuratoren vorgenommen werden, besser zu verstehen. Außerdem zeige ich verschiedene Ansätze, die der besseren Verbreitung von wissenschaftlichen Ergebnissen dienen. Die Reproduzierbarkeit von diesen, oft heterogenen und komplexen, Daten kann mit Hilfe eines innovativen Speicherformats deutlich vereinfacht werden. Ich stelle verschiedene Methoden und Werkzeuge vor, mit denen reproduzierbare Simulationsstudien erstellt, geteilt und gefunden werden können.

Die Lösungen, die ich im Rahmen dieser Arbeit entwickelt habe, existieren dabei nicht nur in der Theorie. Ich konnte viele meiner Ideen in Code gießen. Meine Implementierungen wurden darüber hinaus erfolgreich in andere Anwendungen integriert. Die hier beschriebenen Konzepte fördern das Teilen und Wiederverwenden von wissenschaftlichen Ergebnissen.

THESES

- Models evolve, potentially influencing simulation results. Consequently, it is necessary to track changes. Tracking the evolution of models increases the trust and, thus, fosters reuse of existing models.
- To properly track the evolution of a model, changes between model versions need to be identified. BiVeS is able to detect and communicate the differences between versions of a computational model. Thus, the evolution of a model becomes comprehensible.
- Understanding the changes that occurred in a model is essential for successful collaborations and key to grasp model evolution. Using COMODI, changes can be semantically annotated to communicate reasons, intentions, and effects of an update.
- Changes may be irrelevant to certain users. Annotations with terms from COMODI enable filters for relevant changes in a model.
- Today's simulation studies include multiple, heterogeneous, and sometimes distributed data files, leading to the challenge of exchanging complete and thus reproducible results.
- For simulation studies to be reproducible, researchers need to share all the data and relations between files. Container formats, such as the COMBINE archive, support researchers in sharing their research results.
- Reusable simulation studies require accessible and comparable models and versions thereof.
- Finding a simulation study best suited for a given task is a tough challenge. Tools such as the Cardiac Electrophysiology Web Lab and M2CAT support users in comparing and obtaining reproducible simulation studies.
- Free software supports dissemination of ideas and tools.

CONTENTS

Contents

List of Figures

List of Tables

List of Source Code

List of Abbreviations

1. Reproducibility and reuse of modelling results in the life sciences	1
1.1. The need for reproducibility and reuse	1
1.2. State of the art	5
1.2.1. Standards in the the life sciences	6
1.2.2. Repositories for standardised computational models	10
1.2.3. Model version control	11
1.3. Outline of the thesis	12
2. A method to characterise differences in computational models	15
2.1. The need for comparing models	15
2.2. An algorithm to detect and communicate differences in computational models .	17
2.2.1. Pre-processing the model documents	18
2.2.2. Mapping hierarchical structures	18
2.2.3. Post-processing the mapping	19
2.2.4. Identification of differences and computing of a delta	20

Contents

2.2.5. Translating the delta into machine- and human-readable languages . . .	21
2.2.6. An illustration of the algorithm	22
2.3. Implementing the algorithm for detection and communication of differences . .	24
2.3.1. BiVeS detects and communicates differences between model versions . .	24
2.3.2. BudHat demonstrates the advantages of BiVeS	26
2.3.3. DiViL visualises the differences in standard formats	26
2.4. Semantic characterisation of differences	28
2.4.1. The COMODI ontology	28
2.4.2. Design considerations for the ontology	29
2.4.3. Ontology organisation and content	30
2.4.4. Annotation of changes	32
2.4.5. Prediction of possible consequences of a change	33
2.4.6. Filter for changes	34
2.5. Conclusions and impact	35
3. Applying the method for difference detection in systems biology projects	39
3.1. BiVeS in the wild	40
3.2. Implementations of my method	43
3.2.1. Versioning in SEEK and the FAIRDOM Hub	44
3.2.2. Models and versions in the Physiome Model Repository	46
3.2.3. Comparing models in the Cardiac Electrophysiology Web Lab	48
3.3. Versioning concept for a database	50
3.3.1. Database structure implemented in Masymos	50
3.3.2. Enhanced concept respecting model versions	52
3.3.3. Results and discussion	55
3.4. Studying the evolution of open model repositories	58
3.4.1. Materials and methods	59
3.4.2. Results	62
3.4.3. Example: Changes in the repressilator model	66
3.4.4. Discussion	68
3.4.5. Conclusions	70
4. Support for shareable and reproducible simulation studies	71
4.1. Challenges with the reproduction of modelling results	72
4.2. COMBINE archive and OMEX format	73
4.2.1. Implementation	73
4.2.2. Use cases for COMBINE archives	78

4.3. A fully featured COMBINE archive of a simulation study on syncytial mitotic cycles in <i>Drosophila</i> embryos	80
4.3.1. Materials and methods	81
4.3.2. Data description	83
4.3.3. Data validation and conclusion	85
4.4. Tool support for COMBINE archives	86
4.4.1. The CombineArchive library	86
4.4.2. The CombineArchiveWeb application	87
4.4.3. Generation and execution of reproducible simulation studies using the SED-ML Web Tools	90
4.5. A method to extract reproducible simulation studies from open model repositories	91
4.5.1. Implementation	92
4.5.2. Integration with other tools	93
4.5.3. Summary	94
4.6. Functional curation of simulation studies	94
4.6.1. Materials and methods	96
4.6.2. Exploring model characteristics and discovering steady states	99
4.6.3. Correcting errors in model encodings	99
4.6.4. Impact and discussion	101
5. Conclusion	105
Bibliography	109
A. Issues with LCS and computational models	i
B. BiVeS example with MathML	iii
C. BiVeS' delta format	vii
D. List of short URLs	xi
E. COMODI	xv
E.1. COMODI class hierarchy	xv
E.2. COMODI's representation in Masymos	xvii
F. Demo models used in Masymos	xix
G. Software developed for this thesis	xxv
G.1. BiVeS	xxvi
G.2. BiVeS-StatsGenerator	xxvi
G.3. BudHat	xxvii

Contents

G.4. CaRo	xxvii
G.5. CombineArchive library	xxviii
G.6. CombineArchiveWeb application	xxviii
G.7. CombineExt	xxix
G.8. Functional Curation WebLab	xxix
G.9. jComodi	xxx
G.10.M2CAT	xxxi
G.11.MoSt	xxxi
H. Curriculum vitae	xxxiii
I. Original publications	xxxv
I.1. Papers in refereed journals	xxxv
I.2. Publications in books and proceedings	xxxvii
I.3. Selected talks	xxxviii
I.4. Selected posters	xxxix

LIST OF FIGURES

1.1. A typical modelling workflow	2
1.2. Models and their versions vary in focus and quality	4
1.3. Models evolve over time	5
1.4. Visual outline of the thesis	13
2.1. A model's temporal evolution	16
2.2. Schematic of the mapping procedure	23
2.3. Dependency graph of BiVeS' modules	25
2.4. Output formats generated by BiVeS and BudHat	27
2.5. Development process of COMODI	29
2.6. Structure of the COMODI ontology	33
3.1. A model comparison result in the FAIRDOM Hub	45
3.2. A model comparison result in the Physiome Model Repository	47
3.3. A model comparison result in the Cardiac Electrophysiology Web Lab	49
3.4. Database structure implemented in Masymos	51
3.5. ER diagram of the extension on the database model in Masymos	54
3.6. Representation of a delta in Masymos	57
3.7. Pipeline used to obtain, analyse, and visualise the data for our large-scale study on model evolution	60
3.8. Number and size of models in BioModels Database and in the Physiome Model Repository	62
3.9. Updates of models in BioModels Database	63
3.10. Types of diff operations	64

List of Figures

3.11. Coverage of COMODI terms	65
3.12. Differences between versions of the Repressilator model in BioModels Database	67
4.1. COMBINE archive: One file to share them all	75
4.2. Visualisation of the model in the fully featured COMBINE archive	83
4.3. Comparison of simulation results in the fully featured COMBINE archive	85
4.4. Tool support for COMBINE archives	87
4.5. Screenshot of the CombineArchiveWeb application	89
4.6. Screenshot of the SED-ML Web Tools	90
4.7. The M2CAT workflow	92
4.8. The concept of reusable virtual experiments	95
4.9. Overview of the virtual experiments	98
4.10. Comparison of AP waveforms	100
4.11. Restitution curves for the O'Hara 2011 models	101
4.12. Effect of blockade of NCX on steady-state APD	102
B.1. Typical tools compare MathML	iv
E.1. COMODI' representaiton in Masymos	xviii

LIST OF TABLES

2.1. List of object properties defined in COMODI	32
4.1. Collection of tools, libraries, and databases supporting COMBINE archives	78
4.2. Content of the fully featured COMBINE archive	84
A.1. Comparison of BiVeS and Unix' diff	i
C.1. Content of the fully featured COMBINE archive	ix
G.1. Collection of tools developed for this thesis	xxxii

LIST OF SOURCE CODE

2.1. Annotation of a difference using COMODI	34
3.1. Calling BiVeS through its Java API	40
3.2. Comparing models using the BiVeS web application	41
3.3. Analysing model documents using the BiVeS web application	42
3.4. Calling BiVeS from the command line	43
3.5. Properties of a <code>DIFF_NODE</code> in Masymos	56
4.1. Excerpt of a COMBINE archive manifest	76
4.2. Excerpt of a COMBINE archive metadata file	77
4.3. Query Masymos' annotation index	93
4.4. Query publications in Masymos	93
4.5. Query simulation descriptions in Masymos	93
B.1. MathML encoding of simple formulas	iv
B.2. BiVeS compares MathML	v
C.1. BiVeS' delta example	viii
F.1. Toy models used to demonstrate the versioning concepts on a database layer . .	xx
F.2. Differences between the toy models reported by BiVeS	xxi
F.3. Annotations of the differences between the toy models reported by BiVeS	xxiv

LIST OF ABBREVIATIONS

ACID Atomicity, consistency, isolation, durability.

AP Action potential.

APD AP duration.

API Application programming interface.

ASCII American standard code for information interchange.

ATP Adenosintriphosphat.

BULD Bottom-up lazy-down.

CAT COMBINE archive toolkit.

Cdc2 Cell division control protein 2.

CDK1 Cyclin-dependent kinase 1.

Chaste Cancer, heart and soft tissue environment.

ChEBI Chemical entities of biological interest.

COMBINE Computational modeling in biology.

COMODI Computational models differ.

COPASI Complex pathway simulator.

CSV Comma-separated values.

CVS Concurrent versions system.

D3 Data-driven documents.

de.NBI German network for bioinformatics infrastructure.

DOI Digital object identifier.

EBI European bioinformatics institute.

ELIXIR European life-sciences infrastructure for biological information.

List of Abbreviations

ER Entity relationship.

FAIR Findable, accessible, interoperable, and reusable.

FAIRDOM FAIR data, operating procedures, and models.

FEBS Federation of european biochemical societies.

FTP File transfer protocol.

GNU GNU's not unix.

GO Gene ontology.

HTML Hypertext markup language.

HTTP Hypertext transfer protocol.

IANA Internet assigned numbers authority.

INRIA Institut national de recherche en informatique et en automatique.

ISA Investigation, study, assay.

JAR Java archive.

JSON JavaScript object notation.

JWS Java web simulation.

Kegg Kyoto encyclopedia of genes and genomes.

KiSAO Kinetic simulation algorithm ontology.

LCS Longest common subsequence.

M2CAT Masymos to CAT.

Masymos Management system for models and simulations.

MathML Mathematical markup language.

MIAME Minimum information about a microarray experiment.

MIASE Minimum information about a simulation experiment.

MIRIAM Minimum information required in the annotation of models.

MoSt ModelStats.

MPF Maturation promoting factor.

mRNA Messenger RNA.

MVC Model view controller.

NCX Sodium-calcium exchanger.

NoSQL Not only SQL.

NuML Numerical markup language.

OBO Open biomedical ontologies.

ODF OpenDocument format.

List of Abbreviations

- OMEX** Open modelling exchange format.
- ORCID** Open researcher and contributor ID.
- OWL** Web ontology language.
- PAV** Provenance, authoring and versioning.
- PDF** Portable document format.
- PharmML** Pharmacometrics markup language.
- PLOS** Public library of science.
- PMID** PubMed identifier.
- PMR** Physiome model repository.
- PNG** Portable network graphics.
- PURL** Persistent uniform resource locator.
- R** Programming language and free software environment for statistical computing.
- RDF** Resource description Framework.
- RDF/XML** RDF serialisation as an XML document.
- RDFS** Resource description framework schema.
- REST** Representational state transfer.
- SBGN** Systems biology graphical notation.
- SBGN-AF** SBGN activity flow language.
- SBGN-ED** SBGN process description language.
- SBGN-PD** SBGN entity relationship language.
- SBML** Systems biology markup language.
- SBO** Systems biology ontology.
- SBRML** Systems biology results markup language.
- SED-ML** Simulation experiment description markup language.
- SG** Statistics generator.
- SOP** Standard operating procedure.
- SVG** Scalable vector graphics.
- SVN** Apache Subversion.
- SWT** SED-ML web tools.
- TEDDY** Terminology for the description of dynamics.
- TetR** Tet repressor.
- TURTLE** Terse RDF triple language.
- UniProt** Universal protein database.
- URI** Uniform resource identifier.
- URL** Uniform resource locator.
- URN** Uniform resource name.

List of Abbreviations

UUID Universally unique identifiers.

VCS Version control system.

W3C World wide web consortium.

WAR Web application archive.

WebCAT CombineArchiveWeb application.

XML Extensible markup language.

XPath XML path language.

YOURLS Your own URL shortener.

CHAPTER 1

REPRODUCIBILITY AND REUSE OF MODELLING RESULTS IN THE LIFE SCIENCES

In this chapter I discuss the overall necessity for reproducibility and motivate reuse of modelling results in the life sciences. I touch upon specific objectives of this thesis and provide background information, which helps understanding the work and its impact. The choice of standards and workflows in the present work is closely related to the COMBINE initiative, in which I am actively involved. It will conclude with a graphical outline of the thesis.

1.1. The need for reproducibility and reuse

Modelling and simulation has become a standard tool for research in the life sciences [HP02; Kli+16; Nob08]. The growing impact of *in silico* studies is reflected by a rapidly growing number and complexity of computational models in open model repositories [Hen+10; Li+10; Yu+11]. In addition, research in the life sciences is increasingly collaborative [DP14; OZB08]. Large-scale modelling projects such as the Virtual Physiological Human¹ [Hun06], the global reconstruction of human metabolism² [Thi+13], or the Virtual Liver³ [Hol+12] strongly rely on the availability of reproducible models in standard formats and require techniques for model coupling, merging and combination at different scales. Computational support is needed to manage models; to ensure model exchangeability, stability and validity; and to foster communication between partners [Wal+13].

A central goal of every researcher is to share his findings, ideally in a scholarly publication. Such publications have at least two key roles: (i) to announce a result and (ii) to convince readers that the result is correct [Mes10]. However, scientific achievements are tentative but

¹www.vph-institute.org, accessed 3 May 2017

²humanmetabolism.org, accessed 3 May 2017

³www.virtual-liver.de, accessed 3 May 2017

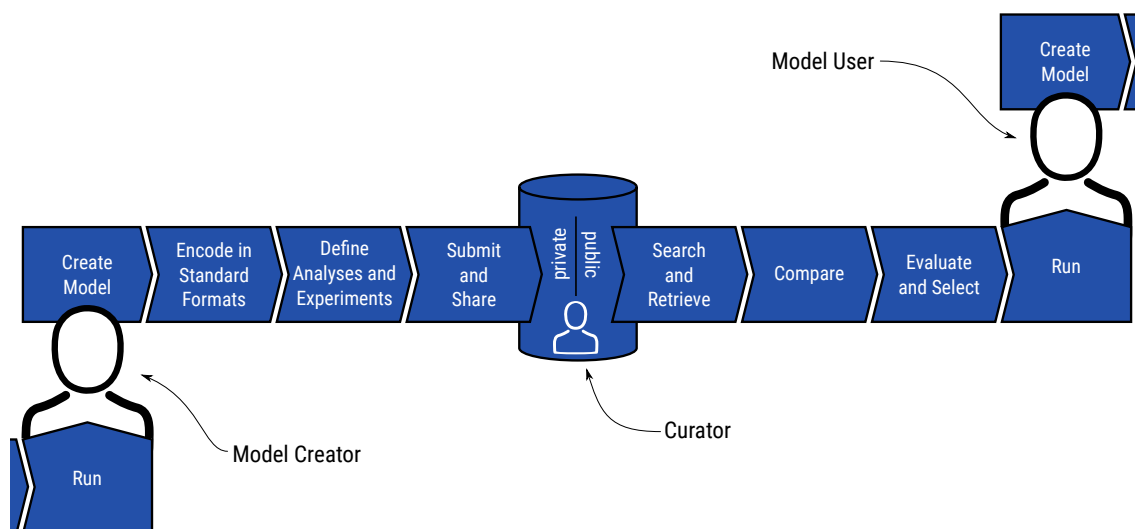


Figure 1.1. A typical modelling workflow. The goal of a *model creator* is to build and publish a model. He creates a model, encodes it in a machine-readable language, defines analyses and runs experiments, before he shares the study with *model users*, for instance through a database. A *model user* needs to obtain a model. He may search in a database, he compares and evaluates candidates, and finally selects a model to test his data and hypotheses. Indeed, the *model user* may also be a *model creator*, if he modifies the retrieved model. Similarly, *model user*, *curator*, and *model creator* may all be the same person, the database being a local directory structure. The figure was derived from [SW16b].

reinforced by reproducibility and reuse [DG10]. In fact, a reused and cited study is clear evidence that the authors managed to persuade their readers that the conclusions drawn are in line with the observations. Thus, reproducibility and reuse of modelling results are vital features in the life sciences [Nos+15], which are not yet widely enough appreciated [WW16]. Figure 1.1 shows a typical modelling workflow, which strongly relies on reusable and reproducible simulation studies. Improving reusability and facilitating sharing accelerate the human aspects of the scholarly knowledge cycle, improve the transfer of know-how, and reduce the “time-to-discovery” [Rou+10]. Research results can be *reused* in different notions [Bec+10]:

- Reuse: Consider the result a “black box” and reuse it as a whole or single entity.
- Repurpose: Substitute services/data of an original study for those used in a derived study.
- Repeat: Run the same study again, perhaps years later.
- Reproduce: Replicate a result, for example to start with the same materials and methods, or to see if a prior result can be confirmed.
- Replay: “Go back and see what happened.” This does not necessarily involve execution or enactment of processes or services, but places requirements on metadata recording the provenance of data and results.
- Trace: Audit the steps performed in experiments and investigations (to be convinced of the validity of results).

Chapter 1. Reproducibility and reuse of modelling results in the life sciences

Each aspect poses a number of challenges. For example, researchers need to share (and find) all the data, used methods, and detailed processes that are relevant to a specific research result [Rou+10]. However, these results are typically very complex and consist of a variety of heterogeneous data. A simulation study in the life sciences may, for instance, depend on (i) a model of a biological system, (ii) some environmental setup, (iii) a simulation protocol, (iv) standard operating procedures, (v) workflows, (vi) semantic annotations, (vii) provenance records, (viii) documentation and a scientific publication, and (ix) simulation results. Similarly, researchers and their biases are not less complex, impeding reproducible research [FCI17]. The scientific literature contains an impressive collection of problems with regard to trust and reproducibility of research results [Bla14; The13], including medical and pharmaceutical studies [Arr11; BE12; Mul11; PSA11], results in technical fields [Bel+09; Ioa+09], in computational biology [Gar+13], in neuroscience [Top+14], and in cancer research [Err+14]. A lack of standards, a lack of quality and quantity of data, a lack of openness, and a lack of transparency have been identified as root causes for irreproducibility of research results [WW16]. As part of this thesis I developed methods and tools that are entailed to the heterogeneous and versatile character of typical simulation studies and support researchers in sharing their research results.

The community fights these issues with computational weapons. For example, standard formats are developed to encode models, environments, protocols and the like. Indeed, reusability is practically impossible without agreement about the formats used to store and exchange the data [Huc+15b]. These formats have several advantages: (i) The same model can be simulated, analysed, and visualised using different software tools; (ii) models encoded in standard formats may outlive the tool used to create the model; (iii) model exchange becomes feasible; and (iv) models can more easily be shared, published, and reused [Sch+16]. Models in standard format can be collected and published in open repositories. These repositories provide a platform for long-term storage, and they add support for model validation, curation and annotation of submitted models. An army of curators ensure quality and reproducibility of published research. The distribution of models through these repositories accelerates collaborative research and encourages model reuse. Thus, open repositories implement the infrastructure necessary to maintain simulation studies. Fostering reinforcement of reproducibility requires inclusion of model source code, data, methods and results in our publications [DG10]. Several publishers already ask for model source code to be made available along with the written paper (including Oxford journals, BMC journals, PLoS journals or the FEBS journal). They recommend upload of model code to open model repositories using standard formats and annotations. In this thesis I analysed publicly available models in open repositories. Moreover, I extended a database with a concept to store versions of models and their differences.

As the community continuously gains new insights into biological systems, associated hypotheses are updated frequently. Thus, developed simulation studies are subject to changes from its creation to curation, publication and later reuse in other contexts [SWW16; Wal+13],

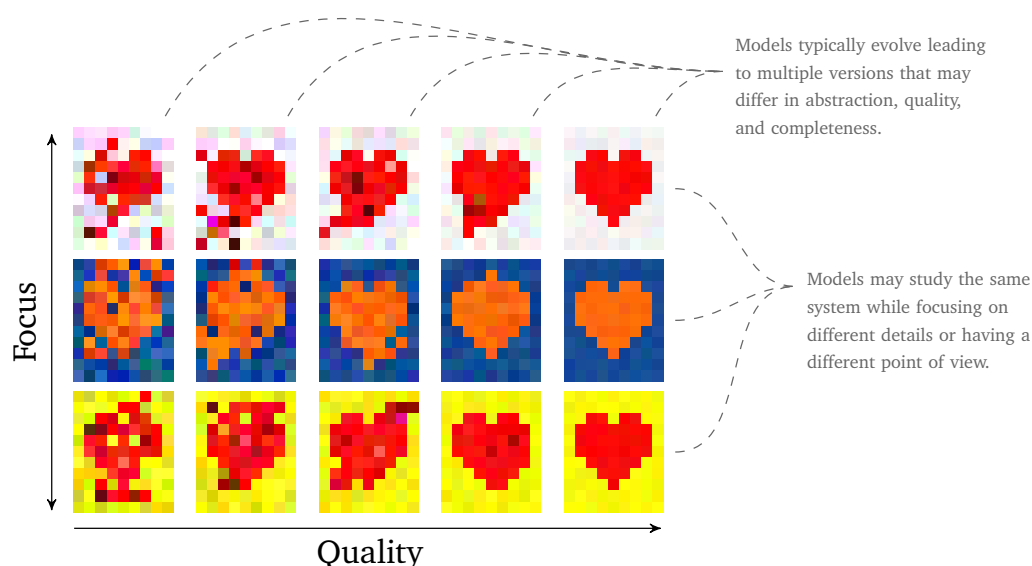


Figure 1.2. Models and their versions vary in focus and quality. The figure sketches three cardiac models (❤️, 🧡, 🟡), which focus on different aspects of the heart under different environments. As models evolve, they typically exist in multiple versions (e.g. 🧡, 🧡, 🧡, 🧡, 🧡), which may differ in quality. The quality of a model version, however, does not necessarily correlate with time.

compare Figure 1.1. The evolution of a model is typically complex, leading to multiple versions that differ in abstraction, quality, and completeness, see Figure 1.2. Changes may involve corrections, extensions, simplifications, and improvements [SWW16]. A reproducible research environment provides computational tools together with the ability to automatically track the provenance of data, analyses, and results and to package them [Mes10]. However, to grasp the evolution of a model all versions need to be available. A version control system (VCS) for models supports users in recapitulating the different modelling steps taken to build a model. This becomes particularly relevant when collaborators or curators need to discuss necessary model changes, e.g., before publication in a model repository. An optimal VCS for models (i) stores all existing versions of a model during its existence, (ii) is able to communicate the differences between versions to both computers and humans, and (iii) allows to filter for specific changes (e.g. “affects reaction network” or “changes kinetics”). Thus, the information recorded by a VCS enables users to study a model’s past and to answer specific questions about the model (compare Figure 1.3). Modellers may investigate which parts of a model have been changed, and how these changes were justified. For example, changes in the model parametrisation may be justified by a new publication. It is also interesting for a modeller to see how often a model has been changed and when it was last changed. This information indicates how recent a model is. Has it been tested? Is it actively used in the community? Positive answers to these questions potentially increase the trust in a modelling result. When reusing model code, which has been changed frequently in the past, the modeller might decide to confirm the model’s validity on a regular basis. Changes in the model may directly affect the simulation results and therefore

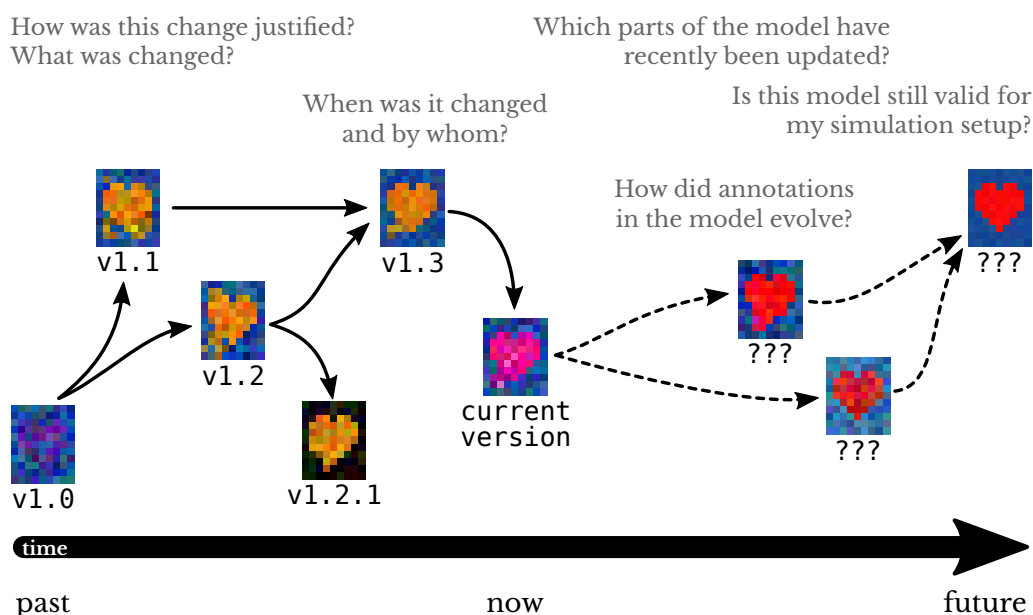


Figure 1.3. Models change over time giving rise to a number of questions. The figure was derived from our previous publication [Wal+13].

must be communicated and explained. In addition, the applicability of standard simulation procedures, termed functional curation [CMN11], must be ensured for each single version of a model. Consequently, the availability of every model version used in a simulation experiment is a major requirement to ensure reproducibility of results. The awareness of a need for version control led to the incorporation of technical solutions in open model repositories. However, previous approaches were appropriate for software code rather than model files. As part of this thesis, I introduced a novel method to identify and characterise differences in computational models. Several applications already implement my method, demonstrating its impact.

1.2. State of the art

Different organisations and approaches already address abovementioned issues [Sta+15]. For example, the Reproducibility Initiative⁴ is a collaboration between Science Exchange, PLOS, figshare and Mendeley, which identifies and rewards high quality reproducible research via independent validation of key experimental results [Pat12; SW15]. Community networks are vital for the scientific domain. The “COMputational Modeling in BIology NEtwork” (COMBINE⁵, [Huc+15b]), for example, coordinates the collaborative development of standard formats. Large-scale projects, such as FAIRDOME⁶, enhance and facilitate collaborations to make research results FAIR: Findable, Accessible, Interoperable, and Re-usable. In FAIRDOME,

⁴validation.scienceexchange.com, accessed 4 April 2017

⁵co.mbine.org, accessed 1 May 2017

⁶fair-dom.org, accessed 16 October 2017

various data management groups develop the required set of tools to extend existing network services for the wider European systems biology community [Wol+17]. Furthermore, funders now require explicit data management strategies in grant applications, and trans-project data management systems such as the SEEK platform have become an integral part of the scientific landscape [Wol+15]. Networks such as de.NBI⁷ and ELIXIR⁸ emerge from massive infrastructure strategies. Various projects work towards reproducible experiments by re-thinking the publication process. For example, Research Objects [Bec+10] support the publication of data, code and other resources. A number of data preservation initiatives, such as the NSF-sponsored DataONE project⁹ ensure the long-term storage and reusability of scientific data on a large scale [Bec+13]. A call for Virtual Experiments highlights the actual benefits of generic simulation setups for the reusability of full behavioural repertoires of computational models [CVW15].

1.2.1. Standards in the the life sciences

Distributed and collaborative modelling is becoming increasingly prominent [DP14]. Basic infrastructure is provided by standard formats. Only standard formats allow for efficient and tool-independent exchange of data and facilitates the exchange and interpretation of the outcomes of scientific research [Kli+07]. Given the diversity of research topics in the life sciences and the different facets of modelling projects, specific standards with particular purposes are needed to formalise and serialise the data [SHL07]. In the following I introduce standards that are relevant for this work.

Encoding models of biological systems

Model representation formats standardise model encoding. Examples are the Systems Biology Markup Language (SBML, [Huc13]) or CellML [Cue+03]. These formats represent a model's structure (e.g. the biochemical network) and allow basic annotation of the model to better convey a model's intention.

SBML is based on XML¹⁰, the specification of SBML is released in levels (major release) and versions (minor release). At the time of writing, SBML Level 3 Version 1 is the current version. Since Level 3 the SBML core language can be extended with packages, which support e.g. visualisations [Gau+13; Gau+15], constraint-based models [BBO10; OB15], hierarchical model composition [Smi+13; Smi+15], or grouping of elements [HS16a; HS16b]. According to the SBML Level 3 specification [Huc+10] a valid model may consist of various user-defined elements. For example, (i) species typically participate in (ii) reactions, either as a substrate, a product, or a modifier. The base mathematical description of the model is formed

⁷denbi.de, accessed 1 May 2017

⁸elixir-europe.org, accessed 1 May 2017

⁹www.dataone.org, accessed 3 May 2017

¹⁰www.w3.org/XML, accessed 20 April 2017

by (iii) functions, (iv) units, (v) parameters, (vi) assignments, and (vii) kinetic definitions. The mathematical system can be enhanced using (viii) constraints, (ix) rules, and (x) events. Every major element carries an identifier; MathML¹¹ is used to encode mathematics. The SBML community developed an annotation scheme [Huc+10] based on the Resource Description Format (RDF, [LS99]) and identifiers from the Minimal Information Required in the Annotation of Models (MIRIAM, [Nov+05]) Registry [JLL12]. SBML is supported by many software tools [SBM17].

Similarly, CellML is an XML-based description language to define models of cellular and subcellular processes. The first stable specification for CellML 1.0 was released in 2001 [HN01]. At the time of writing the current version is CellML 1.1 [Cue+02]. CellML explicitly supports a component-based architecture – models may import (parts of) models – and therefore strongly encourages reuse of models and parts thereof [LHN04; Wim+09b]. A CellML model typically consists of components, which may contain variables and mathematics that describe the behaviour of that component. Units can be defined document-wide, or specifically for certain components. CellML provides means to reuse and group components into hierarchical structures. Similar to SBML, major elements carry identifiers; mathematical definitions are encoded using MathML. A CellML metadata standard allows for biological and biophysical annotation of the models [Bea+09; Wim+09c]. A wide range of software supports CellML¹² [Gar+08; Wim+09a].

Encoding simulation environments and setups

A consistent and standardised description of a model is a prerequisite for its dissemination. However, an additional descriptive layer is necessary to ensure direct result reproducibility. This layer is covered by the Simulation Experiment Description Markup Language (SED-ML, [Wal+11b]), which is a format for the standardised encoding of simulation setups. SED-ML is widely used to exchange simulation experiments in computational biology [Huc+15b]. SED-ML files typically consist of five major blocks of information: Nomination of the model; initialisation of the variables; specification of the simulation algorithm; post-processing of the results; and definition of plots and numerical reports [Ber+15; Wal+11c]. Entities in computational models can be addressed using XPath¹³ expressions. Libraries to read and write SED-ML are provided by the community, and some software tools already consume and export SED-ML files¹⁴, e. g. BioUML [Kol+11; Kol02], COPASI [Hoo+06], JWS Online [OS04], or Tellurium [Sau+16]. Similarly to CellML and SBML, SED-ML allows for annotations of the simulation description [Cou+11; Zhu+12].

¹¹www.w3.org/Math, accessed 20 April 2017

¹²www.cellml.org/tools, accessed 20 April 2017

¹³www.w3.org/TR/xpath, accessed 21 April 2017

¹⁴sed-ml.sourceforge.net/showcase.html, accessed 21 April 2017

Encoding modelling results

Similarly to models and simulation setups, a standard way of communicating the simulation results is necessary. Researchers should record the results of their analysis for validation and subsequent analysis. Results of *in silico* experiments in the life sciences typically include numerical values and figures. Numerical results are usually tables or matrices, encoded in CSV [Sha05]. An early approach to provide structure and schema definition to CSV-like files was fielded text [Kli16]. The Systems Biology Results Markup Language (SBRML, [Dad+10]) is specialised for encoding simulation results obtained by running an SBML model. Using SBRML it is possible to include ontologies for semantic annotations of numerical results. Based on SBRML, the development for the Numerical Markup Language (NuML¹⁵) is currently in progress. LibNUML provides software support for reading, writing and manipulating data in NuML format on all operating systems¹⁶.

Visualising a biological processes is a challenge, especially as everyone draws and understands these figures differently [Kit+05]. The Systems Biology Graphical Notation (SBGN, [Nov+09]) is a standard to encode unambiguous visualisations of modelling results in the life sciences. SBGN comes in three flavours, which allow for different perspectives onto the model: (i) the Process Description (PD) visualises processes, which transform or change entities [Moo+15], (ii) the Entity Relationship (ER) visualises interactions and relationships between entities [Sor+15], and (iii) the Activity Flow (AF) visualises biological activities between entities [Mi+15]. For every flavour, a standardised set of entities (glyphs) and relations visualises the model in a so-called map. A standardised markup language (SBGN-ML) can be used to independently encode the map [Ier+12].

Encoding semantics

Sustainable model reuse requires a basic understanding of (i) the biological background, (ii) the modelled system, and (iii) possible parametrisations under different conditions [Sch+16]. The knowledge about a modelled system can be transferred using semantic annotations. Thus, the semantic layer promotes reuse of the modelling project [Mis+16]. For this purpose, terms from ontologies and controlled vocabularies can be linked to the entities of a project using semantic technologies, such as RDF [LS99; RDF14]. An ontology is a tool to provide meaning to data, the information of which can then be subjected to algorithmic processing [Ash+00; Smi+07]. It clarifies the intended semantics of the data, which makes the data more accessible, shareable, and interoperable [Gen+11]. Bio-ontologies are formal representations of knowledge in biology [BR04; RM03]. For example, the Gene Ontology [Ash+00] provides additional information on the genomic level, the NCBI Taxonomy [Fed12] provides information about the nomenclature of species, UniProt [The07] provides information about proteins, and

¹⁵github.com/numl/numl, accessed 1 May 2017

¹⁶github.com/NuML/NuML/tree/master/libnuml, accessed 1 May 2017

Chapter 1. Reproducibility and reuse of modelling results in the life sciences

ChEBI [Deg+07] provides information about chemical entities of biological interest. Using the vCard ontology [IM14; Per11] it is possible to relate to people and organisations. The following ontologies [Cou+11] are especially important for this work:

- the Systems Biology Ontology (SBO¹⁷) designed for models in systems biology,
- the Kinetic Simulation Algorithm Ontology (KiSAO¹⁸) built for simulation descriptions,
- the TErminology for the Description of DYnamics (TEDDY¹⁹) designed for dynamical behaviours and results.

Machine-readable annotations automate exchange, validation, reuse, and composition of models. The semantic layer can also be exploited to convert machine-readable code into human-readable formats, such as PDF documents [Drä+09; SBS10] or visualisations [Jun+12; Roh+12] to aid human understanding.

Several guidelines support researchers in annotating their scientific results. For example, the initiative “Minimum Information for Biological and Biomedical Investigations” (MIBBI, [Ket+10; Tay+08]) provides a general checklist for results in the domain of the life sciences. More specifically, the guideline on “Minimum information requested in the annotation of biochemical models” (MIRIAM, [Nov+05]) provides a checklist specifically entailed for computational models. According to MIRIAM, the description of a computational model requires

- a valid implementation in an appropriate language,
- an initial parametrisation,
- proper metadata about its provenance (creators, contributors, and creation/modification dates, terms of distribution etc.) and references to a corresponding publications or similar documentations,
- the reproducibility of the references publication/documentation.

Similarly, the guideline on “Minimum Information About a Simulation Experiment” (MIASE, [Wal+11a]) provides a checklist for simulation descriptions. According to MIASE, a simulation experiment needs to specify

- a comprehensive model including equations and parametrisations,
- a simulation description including precise description of the simulation steps,
- anything that is necessary to obtain described results.

Other minimum-information initiatives include guidelines for the description of microarray experiments (MIAME, [Bra+01b]), genome sequence (MIGS, [Fie+08]), proteomics experiments (MIAPE, [Tay+07]), and cardiac electrophysiology experiments (MICEE, [Qui+11]).

¹⁷www.ebi.ac.uk/sbo/main, accessed 2 May 2017

¹⁸co.mbine.org/standards/kisao, accessed 2 May 2017

¹⁹co.mbine.org/standards/teddy, accessed 2 May 2017

1.2.2. Repositories for standardised computational models

Open repositories collect and publish curated models and related data [Wal+13]. They provide the necessary infrastructure (i) to exchange and access modelling projects with a rich set of model-related information [Hen+10], (ii) to maintain, curate, and validate model code and associated metadata [SWW16; Wal+13], and (iii) to track and archive published (versions of) modelling projects [Wol+11]. Thus, creating and populating model repositories requires expert knowledge and integration of heterogeneous data from various sources [Mis+16]. The distribution of models through open repositories accelerates collaborative research and encourages model reuse [SWW16]. Publishers encourage modellers to share model source code along with the written paper. They recommend upload of model code to open model repositories using standard formats and annotations. Consequently, open repositories are high-quality resources of ready-to-reuse models [Wal+13]. Common repositories for simulation studies in the life sciences include BioModels Database [Li+10], the Physiome Model Repository [Mil+11; Yu+11], SEEK [Wol+17], JWS online [vGen+07], ModelDB [Hin+04; McD+15] or the Open Source Brain²⁰. This work will focus on the first three of this list.

BioModels Database is a repository of freely accessible models encoded in SBML format. New models are submitted to the non-curated branch. Once the modelling results could be reproduced by a curator, the model moves to the curated branch. BioModels Database tracks the versions of model files using Subversion²¹ (SVN). The BioModels Database team releases a version of their database sporadically about once a year. This way, researchers get rudimentary access to the history of a model. However, changes that occur between two releases are invisible for the users [Wal+13]. At the time of writing, there are 31 public releases. Release 31 of BioModels Database contains 640 models in the curated branch and 1000 models in the non-curated branch.

The Physiome Model Repository provides curated and non-curated models, primarily in CellML format. The models are embedded in workspaces, which may contain further model-related data, such as network visualisations, simulation descriptions, and links to previous versions of the studies. Particularly interesting and well documented revisions of workspaces can be published as exposures. The Physiome Model Repository implements a Git²² system. Thus, its users have access to all the versions of a model and get basic provenance information through the revision control log. At the time of writing, there are 2782 models files in 651 publicly available workspaces.

In contrast, SEEK is not only a model repository but a management platform designed to organise all kinds of data in systems biology. SEEK implements the Investigations-Studies-Assays (ISA) structure, which makes the model construction and validation transparent [San+08;

²⁰ opensourcebrain.org, accessed 3 May 2017

²¹ subversion.apache.org, accessed 2 May 2017

²² git-scm.com, accessed 2 May 2017

[San+12](#)]. A sophisticated user and permission system controls the visibility of projects and entities. Users can work privately or collaborate in groups, and they may share their work with the public. SEEK is open source and can be installed by everyone. The FAIRDOMHub²³ provides a public and curated SEEK instance.

Models in the life science often encode networks of biological processes. The Masymos database [[HWW15](#)] is a young approach to store these networks in a graph database (Neo4J²⁴). Indeed, a graph database is well suited to store networks and can be used to link model related information, such as annotations, publications, or persons. This ultimately provides an infrastructure for sophisticated network-based search queries [[Hen+10](#)].

1.2.3. Model version control

As Figures 1.2 and 1.3 suggest: Models evolve over time [[SWW16](#)]. The need for model version control has been previously discussed in research groups facing model evolution in computational biology [[Bea+09](#); [CNH05](#); [Huc+10](#); [Li+10](#); [Mil+11](#); [SO09](#); [Wal+13](#)]. Before I started this work, all applied version control relied on systems such as SVN, Git²⁵, or Mercurial²⁶. In general, VCSs track every change made in a source document, along with information about the author of a change and an optional log message containing information on the reason of a change. However, those systems are effectively using Unix' diff utility to detect changes between versions of a document, which is based on solving the longest common subsequence (LCS) problem [[HM76](#)]. However, this line-based approach to compare versions performs poorly on computational models, because models usually encode networks or hierarchical structures. LCS does not respect the XML structure [[RSB05](#)] that constitutes model files. A model file may already change when a model is loaded and subsequently exported by a simulation software. Even though the model itself did not change, the indentation of the XML code or the order of XML elements may have changed, leading to many irrelevant modifications reported by Unix' diff tool. Model code is often automatically generated in software tools. Each software tool has its own preferred way of representing the model code, sorting the occurring XML elements or breaking lines in the XML code. These common changes are detected by the LCS algorithm, but they do not affect the encoded model. In other words, although being successfully applied to version control of source code, Unix' diff is not suitable for XML version control [[Cha+96](#)]. See Appendix A for more details.

Recently, sbml-diff was published [[SP16](#)]. Using sbml-diff it is possible to visually compare the reaction networks encoded in SBML models. Both networks are mapped onto each other and exported in the Dot²⁷ format, which can easily be compiled into a coloured image using the

²³fairdomhub.org, accessed 2 May 2017

²⁴neo4j.com, accessed 2 May 2017

²⁵git-scm.com, accessed 3 May 2017

²⁶mercurial-scm.org, accessed 3 May 2017

²⁷graphviz.org/content/dot-language, accessed 2 May 2017

GraphViz²⁸ software [Ell+01]. However, modifications in the parametrisation, units, or the mathematical definition of a model are difficult to understand.

1.3. Outline of the thesis

My thesis is organised in five chapters, including this introductory [Chapter 1](#) and a [Chapter 5](#), which concludes this work. In [Chapter 2](#), I describe a method to identify, characterise, and communicate changes between versions of computational models (see top of [Figure 1.4](#)). [Chapter 3](#) shows how I integrated my tools with existing systems biology projects, and I investigate the evolution of open model repositories (see bottom of [Figure 1.4](#)). In [Chapter 4](#), I present methods and tools that support researchers in sharing their research results (see middle of [Figure 1.4](#)). An Appendix contains supplementary material. I published most of this work in refereed journals and on conferences. Corresponding articles are referred to in short synopses at the beginning of every chapter.

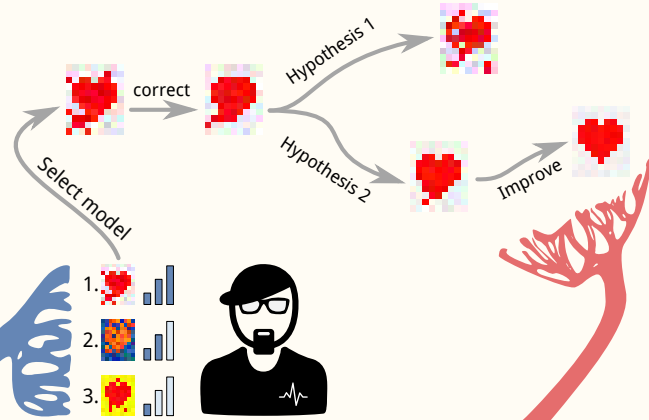
[Figure 1.4](#) illustrates the outline based on a fictive story of a modelling project in systems biology. A researcher developing cardiac models searches in an open model repository for models that would help him studying his data. The models that he retrieves vary both in quality and focus. The researcher selects a model to reuse in his own study. However, he then stumbles upon an error in the obtained model and corrects it. Based on the corrected model, he investigates different hypotheses derived from his data. He may extend or improve the model, before he eventually shares his findings with the community. Thus, he uploads the new model to the repository and shares the results with his social network.

²⁸www.graphviz.org, accessed 2 May 2017

In search for the perfect heart.

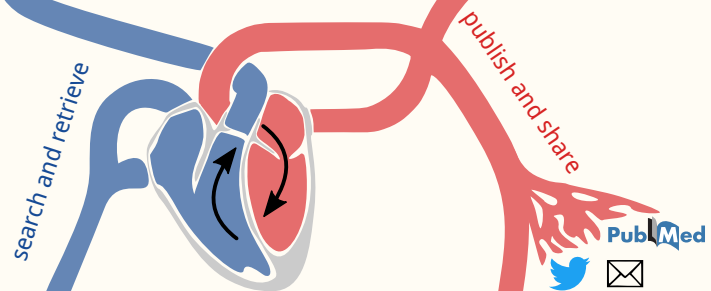
Chapter 2: A method to characterise differences in computational models

The development of a model typically results in several versions of that model. Chapter 2 shows tools to understand the changes and to grasp the evolution of a model.



Chapter 4: Sharing reproducible simulation studies

Ensuring reproducibility when retrieving or publishing a simulation study is a challenge. Chapter 4 presents approaches to support users in sharing research results.



Chapter 3: Application of difference detection in systems biology projects

It is difficult to managing simulation studies and versions thereof. Chapter 3 presents how my tools are used with open repositories to support the modellers.

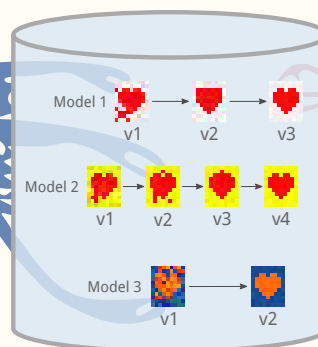


Figure 1.4. Visual outline of this thesis. The figure shows a researcher, who is interested in cardiac models. A model repository contains a number of cardiac models (■, ♥, ♦) in different versions (e.g. ♦, ♦, ♦, ♦) varying both in focus and quality. He retrieves a few models, decides for one of them, creates multiple versions while working with that model, eventually publishes his updated model in the model repository and shares his findings with his social networks.

CHAPTER 2

A METHOD TO CHARACTERISE DIFFERENCES IN COMPUTATIONAL MODELS

Reusable simulation studies require accessible and comparable models and versions thereof. Model provenance and version control enable the widespread use and application of models, saving time and efforts during development. Understanding the changes that occurred in a model is essential for successful collaborations and key to grasp model evolution. Tracking the evolution of a model significantly increases the trust in a model and, thus, fosters reuse and extension of existing models.

This chapter is based on three refereed publications. I published a novel method for detecting and communicating the differences in models of biological systems as a conference paper at the International Conference on Data Integration in the Life Sciences [SWW14] and as an original paper in Bioinformatics [SWW16]. To semantically characterise these differences, I developed an ontology. This ontology was published in the Journal of Biomedical Semantics [Sch+16].

2.1. The need for comparing models

New versions of a model are regularly being generated leading to a potentially complex history of that model. Indeed, models are subject to modifications, including corrections, extensions, and other refinements [CLN13]. During my time as a doctoral candidate I studied and induced the evolution of many models. For example, in 1993, Novak and Tyson published the first cell cycle model describing the M-phase control in *Xenopus oocyte* extracts and intact embryos [NT93]. The model, which encodes these findings, was first published in BioModels Database in 2007 (release number 8) and received the identifier BIOMD0000000107¹. Along with many official releases of BioModels Database, the model has undergone numerous changes. However, changes between 2007 and 2013 only reflect regular updates of the internal encoding of the

¹identifiers.org/biomodels.db/BIOMD0000000107, accessed 3 May 2017

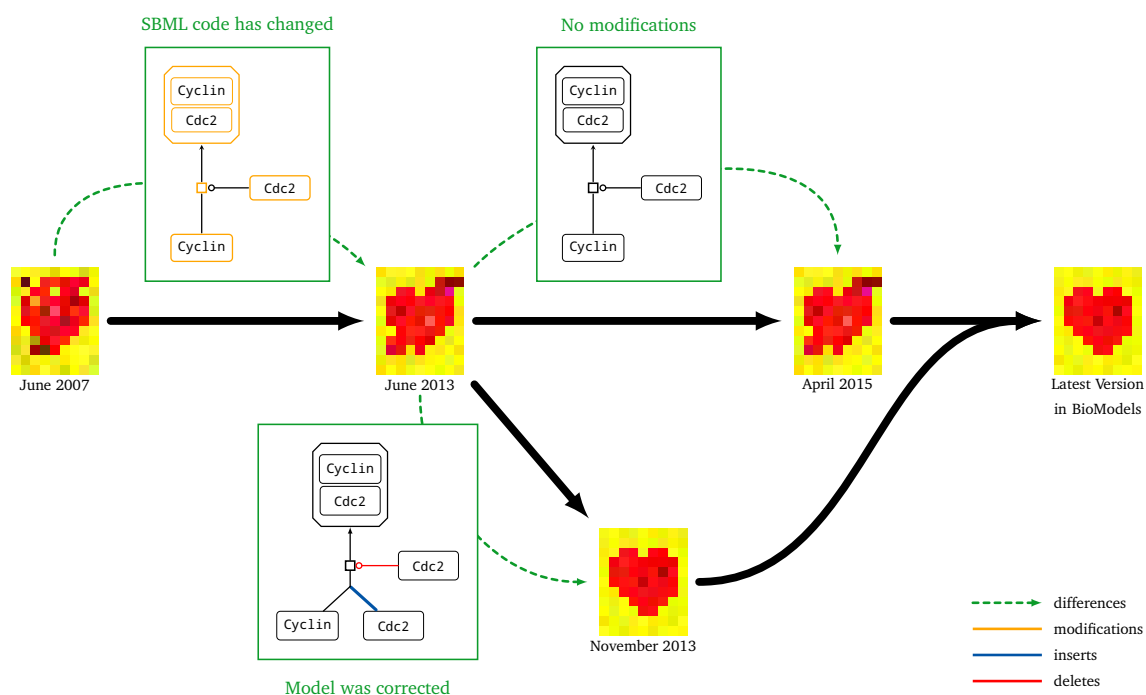


Figure 2.1. Sketch of a model's temporal evolution. Changes in a single reaction of Novak and Tyson's model with ID BIOMD000000107 in BioModels Database. The figure shows the differences between versions from June 2007 (release number 8), June 2013 (release number 25), and April 2015 (release number 29). The branch represents my modification in November 2013. The green boxes visualise the differences between related versions in SBGN format. As of early 2016, the maintainers of BioModels Database merged the corrected version into their datasets (release number 30). Please note that the indicated timeline is not linear.

model in BioModels Database. These modifications do not relate to the biological meaning of the reaction. In November 2013, I detected an error in the model encoding, corrected it, and sent it back to BioModels Database. In early 2016 (release number 30) my corrections were merged into the database. Figure 2.1 visualises that example. It shows the differences in one of the model's reactions: The formation of Cdc2-cyclin dimers from Cyclin subunits and free Cdc2 monomers (Step 3 in [NT93]).

In 2013 we defined major requirements of a comprehensive version control system for computational models: (i) it must be entailed to the structure of model documents, (ii) changes must be transparent and versions must be unambiguously identifiable, addressable, and accessible, (iii) changes must be justified [Wal+13]. However, only with difference detection at hand users are able to grasp a model's history and to identify errors and inconsistencies. Therefore, a framework to detect, characterise, and communicate the differences between models and their versions is a fundamental requirement to compare and combine models.

The following sections deal with versions of XML-based documents in detail. Unless explicitly stated otherwise, I refer to the original document tree as T_1 and to the modified document tree as T_2 . Example nodes from the original document T_1 are called m , while nodes from T_2

are called n . I use other lower-case letters if I need more than two nodes for an example. To simplify the explanation I define the following function on XML trees:

- $\text{root}(T)$ returns the root node of the XML document T .
- $\text{nodes}(T)$ returns all nodes in an XML document T .
- $\text{children}(x)$ returns a set of nodes, which are children of node x .
- $\text{parent}(x)$ returns the parent node of x . It is undefined for the root node of a document.
- $\text{siblings}(x)$ returns the set of nodes, which have the same parent as x .
- $\text{level}(x)$ returns the number of ancestors of x . $\text{level}(x)$ equals 0 if x is the root node.
- $\text{ancestors}(x)$ returns a set of all ancestors of x in the document tree up to the root node.
- $\text{ancestors}(x, i)$ returns a set of all ancestors of x in the document tree up to the i -th ancestor of x (or up to the root node if $\text{level}(x) < i$).
- $\text{length}(x)$ returns the length (number of characters) of the text in x . This operator is exclusively allowed for text nodes in the XML tree.
- $\text{id}(x)$ returns an abstract identifier for node x . Here, the `id` attribute of a document node and its semantic annotations are used to identify a node.
- $\text{attributes}(x)$ returns the set of attributes defined in x .
- $\text{value}(x, y)$ returns the value of attribute y in node x .

In the following Section 2.2, I present my novel algorithm for difference detection in models of biological systems. Section 2.3 introduces BiVeS, BudHat, and DiVil, a tool chain that implements the algorithm. Afterwards, Section 2.4 demonstrates how changes can be characterised semantically using the COMODI ontology. Together, these tools help to identify and characterise differences between computational models, and thereby contribute to the documentation of a model's history, as shown in Chapter 3.

2.2. An algorithm to detect and communicate differences in computational models

My algorithm for detection and communication of differences compares two versions of an XML-encoded model. It concentrates on efficient and reliable difference detection and runs in five major steps: Pre-processing the model documents (Section 2.2.1), mapping hierarchical structures (Section 2.2.2), post-processing the resulting mapping (Section 2.2.3), identification of differences and computing of a delta (Section 2.2.4), and translating the delta into machine- and human-readable languages (Section 2.2.5). All steps are described in technical detail in the following sections. Sections 2.2.1, 2.2.2, and 2.2.4 follow the original ideas of the XyDiff algorithm which was developed at the French Institute for Research in Computer Science and Automation (INRIA) and focuses on efficiency in terms of speed and memory [CAH02].

2.2.1. Pre-processing the model documents

First, two versions of an XML-encoded model are translated into an internal tree structure. For every node x in the tree a hash sum x_σ and a weight x_ω are calculated. The weight x_ω is determined as proposed in the original XyDiff algorithm [CAH02]:

$$x_\omega = \begin{cases} 1 + \log(\text{length}(x)) & \text{if } x \text{ is text node,} \\ 1 & \text{if } x \text{ is leaf,} \\ 1 + \sum_{c \in \text{children}(x)} c_\omega & \text{otherwise.} \end{cases}$$

The weight of a node is thus always greater than the weight of its children:

$$x \in \text{ancestors}(y) \Rightarrow x_\omega > y_\omega$$

As such, the weight represents the size of the corresponding subtree.

The hash sum of a node x represents the signature of the subtree rooted at x . In the current version of my implementation, I determine the hash x_σ of a node x by calculating the SHA-2 sum over a concatenation of (i) the node's tag name, (ii) its attributes, and (iii) the hash sum of all its children. While x_σ unambiguously identifies the subtree rooted in x , x_σ is not necessarily unique in a tree: Identical subtrees will result in the same signature. However, if $x_\sigma = y_\sigma$ then the subtrees in x and y are identically equal.

2.2.2. Mapping hierarchical structures

To compare two XML tree structures T_1 and T_2 , I use XyDiff's BULD algorithm, in which mappings are propagated *Bottom-Up* and only *Lazily Down*. The algorithm finds mappings between common large subtrees of the two documents and propagates these mappings into the rest of the documents [CAH02]. I distinguish the following four phases of the mapping.

Mapping by ID: First, nodes are being mapped with respect to their identifiers. As suggested in the original algorithm, the `id` attributes in the XML documents serve as identifiers. These attributes are efficient indicators but may be meaningless and misleading. In addition, I also evaluate biological identifiers, specifically links into bio-ontologies. In my current implementation, I assign a higher priority to biological identifiers than to `id` attributes. In this step, nodes in both documents which share the same identifier are mapped onto each other:

$$\text{id}(m) = \text{id}(n) \Rightarrow m = n \quad m \in \text{nodes}(T_1), n \in \text{nodes}(T_2)$$

If many nodes are labelled with an `id` attribute and if the model is well annotated, then a large number of mappings are already computed at this stage. Consequently, the following mapping procedure, which is computationally harder, significantly simplifies.

Chapter 2. A method to characterise differences in computational models

Bottom-Up propagation: Second, the initial mapping is propagated upwards into the trees. The connections of a node's children are evaluated in a depth-first traversal of T_2 . If a node n in T_2 is connected to a node m in T_1 then a mapping of $\text{parent}(n)$ to $\text{parent}(m)$ is suggested. The confidence equals n_ω and is therefore proportional to the size of n 's subtree. If, in contrast, n is not connected, the candidates that are suggested by the connections of n 's children are examined. Candidates which have a different tag name than n and candidates which already have a connection are immediately neglected. Among the remaining candidates, the algorithm chooses the one that received the best suggestions (most confidence) and connects it to n .

Top-Down propagation: Third, the algorithm makes use of the initially computed signatures and maps nodes of T_2 on nodes of T_1 which share the same hash value. A priority queue Φ is maintained to sort the nodes of T_2 based on their weights. Initially, Φ only contains the root node of T_2 . Unless Φ is empty, the algorithm repeatedly removes node n :

$$n \mid n_\omega \geq o_\omega \wedge n \neq o \wedge n, o \in \Phi \subset \text{nodes}(T_2)$$

which represents the biggest subtree in the queue, and collects a set of mapping candidates $M \subset \text{nodes}(T_1)$ with $\forall m \in M : m_\sigma = n_\sigma$. If M is empty all children of n are added to Φ and the loop continues with the next biggest subtree in Φ . Otherwise, the algorithm tries to find a node $m \in M$ for which a mapping between nodes x and y with $x \in \text{ancestors}(m, i)$ and $y \in \text{ancestors}(n, i)$ already exists. As proposed by [CAH02] the number of levels i to chase the ancestry of both nodes depends on the ratio of n_ω to $\text{root}(T_2)_\omega$. Thus, for large subtrees i is large and the algorithm climbs many levels in the tree to find a mapping of ancestors, but it might just examine first-hand parents in order to map leaf nodes. If there is an m that meets these conditions, all nodes of the subtrees in m and n are mapped onto each other, just as their ancestors up to the discovered mapping (as long as they share the same tag name).

Optimisation: Fourth, the algorithm improves the quality of the mapping by examining the network structure of T_1 and T_2 in a top-down approach. For every mapping $n \in \text{nodes}(T_2)$ on $m \in \text{nodes}(T_1)$ it compares unmatched children of n and m in order to find missed mappings. A distance matrix $M^{|\text{children}(n)| \times |\text{children}(m)|}$ is created with $M_{i,j}$ being the ratio of the number of differing attributes to the total number of attributes between the i -th child of n and the j -th child of m , or 0 if both nodes do not have any attributes. The algorithm assigns ∞ to elements $M_{i,j}$ if the corresponding nodes already have a mapping, or if they do not share the same tag name. It then evaluates the matrix greedily and adds new mappings up to a maximum distance of 0.9. Thus, nodes which have nothing in common will not be connected.

2.2.3. Post-processing the mapping

Additional mapping rules capture the domain characteristics of the processed data. Following the current specifications for SBML and CellML, I prohibit certain changes in the hierarchical

Chapter 2. A method to characterise differences in computational models

tree of document nodes. Specifically, I treat parts of the model as atomic constructs for which I define restrictions on possible network operations. In SBML models, for example, `listOf`-nodes must not change their parents. That means, if a `listOfModifiers` of T_1 is mapped onto a `listOfModifiers` of T_2 but their parents are not linked, then this mapping is dropped. Similarly, nodes with tag names `speciesReference`, `trigger`, `eventAssignment`, `modifierSpeciesReference`, `delay` and `priority` are glued to their respective parents. If the parents in the corresponding tree are not connected, which means their networks in the XML documents differ, the mapping is removed. In CellML models, for example, nodes with tag names `variable` and `reaction` are glued to their components. Obviously, these rules expand the set of operations in the delta later on, but I deliberately trade some minimality to increase the significance of produced deltas.

2.2.4. Identification of differences and computing of a delta

Here, a delta is a set of operations on XML entities (nodes or attributes, respectively) necessary to transform one document into another [Hel+02; Rei91; Tic85]. I distinguish the following four types of operations which are applied to entities in the corresponding XML tree:

- insert** if an entity is present in T_2 but absent in T_1
- delete** if an entity is present in T_1 but absent in T_2
- move** if a node is present in both documents, but (i) the parents in the corresponding trees are not connected or (ii) the sequence of their siblings has changed
- update** if an attribute's value, a text node's content, or the tag name of a node was modified

While move operations exclusively affect document nodes, update operations typically only change the content of text nodes or attributes. There is a single exception: The root nodes of both documents are always mapped onto each other. I therefore need to support the update of a document node, which changes its tag name. However, this operation is only supported for root nodes. Thus, internal document nodes will never occur in the set of updates.

After the mapping I distinguish two types of nodes: Mapped nodes and unmapped nodes. Unmapped nodes $x \in \text{nodes}(T_1) \cup \text{nodes}(T_2)$ are nodes for which the algorithm could not find a matching node in the opposite tree. These nodes and their attributes correspond to either inserts or deletes, depending on their origin (T_2 or T_1 , respectively). In contrast, mapped nodes are nodes for which the algorithm did find a matching node in the opposite tree. If the parents of such a mapping of $n \in \text{nodes}(T_2)$ onto $m \in \text{nodes}(T_1)$ are not connected, or if the sequence among their siblings has changed, then these nodes are included in the set of moves. Moreover, for each attribute $a \in \text{attributes}(n) \cup \text{attributes}(m)$:

- if $a \notin \text{attributes}(m)$ then a is included into the set of deletes
- if $a \notin \text{attributes}(n)$ then a is included into the set of inserts
- if $\text{value}(m, a) \neq \text{value}(n, a)$ the attribute is included into the set of updates

All other cases (nodes are mapped and occur at the same position in both trees; attribute values of mapped nodes are equal) do not call for an operation to transform T_1 into T_2 and are therefore not included in the delta.

2.2.5. Translating the delta into machine- and human-readable languages

To communicate the identified differences, the delta needs to be serialised. For a machine-readable transmission, the resulting delta is encoded in an XML document. The document consists of the four sections *deletes*, *inserts*, *moves* and *updates*. These sections contain three types of nodes:

- Nodes with a tag name `node`, describing operations on document nodes
- Nodes with a tag name `attribute`, describing operations on attribute values
- Nodes with a tag name `text`, describing operations on text nodes

All these nodes have to carry a unique `id` attribute and, if available, must contain identifiers `oldPath` and `newPath` to unambiguously point to the corresponding nodes in T_1 and T_2 , respectively. These identifiers are XPath² expressions, a language defined by the World Wide Web Consortium (W3C), to identify nodes in an XML document. In addition, `node` nodes may also contain the attributes:

- `oldParent` and `newParent` (XPath expressions), pointing to parents of corresponding nodes
- `oldChildNo` and `newChildNo` (Integers), defining the position among their siblings, in order to encode *moves*
- `oldTag` and `newTag` (Strings), specifying the tag name of the corresponding nodes

Furthermore, `attribute` nodes may have three additional attributes:

- `name`, defining the name of the corresponding attribute
- `oldValue` and `newValue`, specifying the value of that attribute in T_1 and T_2

The generated delta is complete and, thus, it is invertible. That means, it contains all information necessary to transform T_1 into T_2 , but it can also be used to obtain T_1 given T_2 . Figure 2.4c shows an example of an XML-encoded delta. More information about the delta format including a schema definition and further examples can be found in Appendix C.

To support the readability by users, the delta can be exported in two different human-readable formats: A text-based report and a graphical representation of the reaction network. The text-based report lists all modified entities relevant for the biological model (such as parameters, species and reactions) and details on the changes. The report can be generated in HTML³, ReStructuredText⁴, or Markdown⁵. Markdown and ReStructuredText are easy-to-read plain-

²www.w3.org/TR/xpath, accessed 31 May 2017

³www.w3.org/html, accessed 31 May 2017

⁴docutils.sourceforge.net/docs/ref/rst/restructuredtext.html, accessed 31 May 2017

⁵daringfireball.net/projects/markdown, accessed 31 May 2017

text markup languages, specifically designed to ensure a straightforward conversion to other markup languages, such as (X)HTML, doc(x), ODF, or \LaTeX .

If the analysed models contain information about the encoded reaction network, the differences can be visualised in a graph. Therefore, the extracted reaction networks of T_1 and T_2 are translated into an internal graph representation. More specifically, the network of T_2 is stacked on top of the network of T_1 just like an overlay, according to the previously computed mapping between both documents. Subsequently, the combined graph is evaluated: The algorithm checks whether nodes and edges originate from one or both documents and analyses what has changed in the corresponding tree nodes. To export this graph, I developed translators that convert the internal graph representation into exchangeable graph formats, such as GraphML⁶ or Dot⁷, see Section 2.3.1. These graphs can then be visualised in end-user applications.

2.2.6. An illustration of the algorithm

Figure 2.2 exemplifies the algorithm. It shows two versions of a minimalist model, following the SBML structure. Here, the reaction $C + D \rightleftharpoons E$ (left) is updated to $D + H \rightleftharpoons E$ (right). My method first transforms the model files into internal tree representations and prepares the trees for the subsequent mapping procedure (row one, *pre-processing*). The weights ω of nodes in the tree are computed according to the size of the corresponding subtrees. For example, the subtree rooted in B is larger than the subtree rooted in F and, thus, B's weight is greater than F's (namely $B_\omega = 4$ and $F_\omega = 2$, respectively). The mapping procedure starts in row two of Figure 2.2 with a *mapping by id*. Since the *id* attribute plays a key role and many elements do carry *id* attributes, the algorithm typically finds a large number of mappings at this early stage. In this example, only the identifiers of the G-nodes are identical (*id*="reaction1") and thus only a single connection is found.⁸ The *mapping by id* phase is followed by a *bottom-up propagation* (row three), which makes use of the parent-child relation of nodes in the trees: For nodes that are mapped already, there is a good chance that their parents also stem from each other. In the example, the mapping of the G-nodes is propagated towards the roots of the trees and the A-F-G-paths in both model versions are mapped. Afterwards, the algorithm tries to map subtrees with an equal signature (row four, *top-down propagation*). The signatures σ , which are computed in the *pre-processing* step, uniquely identify the subtrees. Here only the signatures of the D-nodes are equal ($D_\sigma = x$ in both versions), which is why D is the only candidate for a mapping. Since the D-nodes originate from each other, as well as the A-nodes do, a mapping of the B-nodes is added (B is child of A and parent of D in both documents). Following the propagation phases, the algorithm tries to connect unmapped children of mapped nodes (row five, *optimisation*). In this example, only the B-nodes have unmapped children: Nodes C and E in version 1 and nodes E and H in version 2 are not mapped, yet. To find a mapping

⁶graphml.graphdrawing.org, accessed 31 May 2017

⁷graphviz.org/content/dot-language, accessed 31 May 2017

⁸For demonstration purposes I assume that the D-nodes do not carry *id* attributes.

Chapter 2. A method to characterise differences in computational models

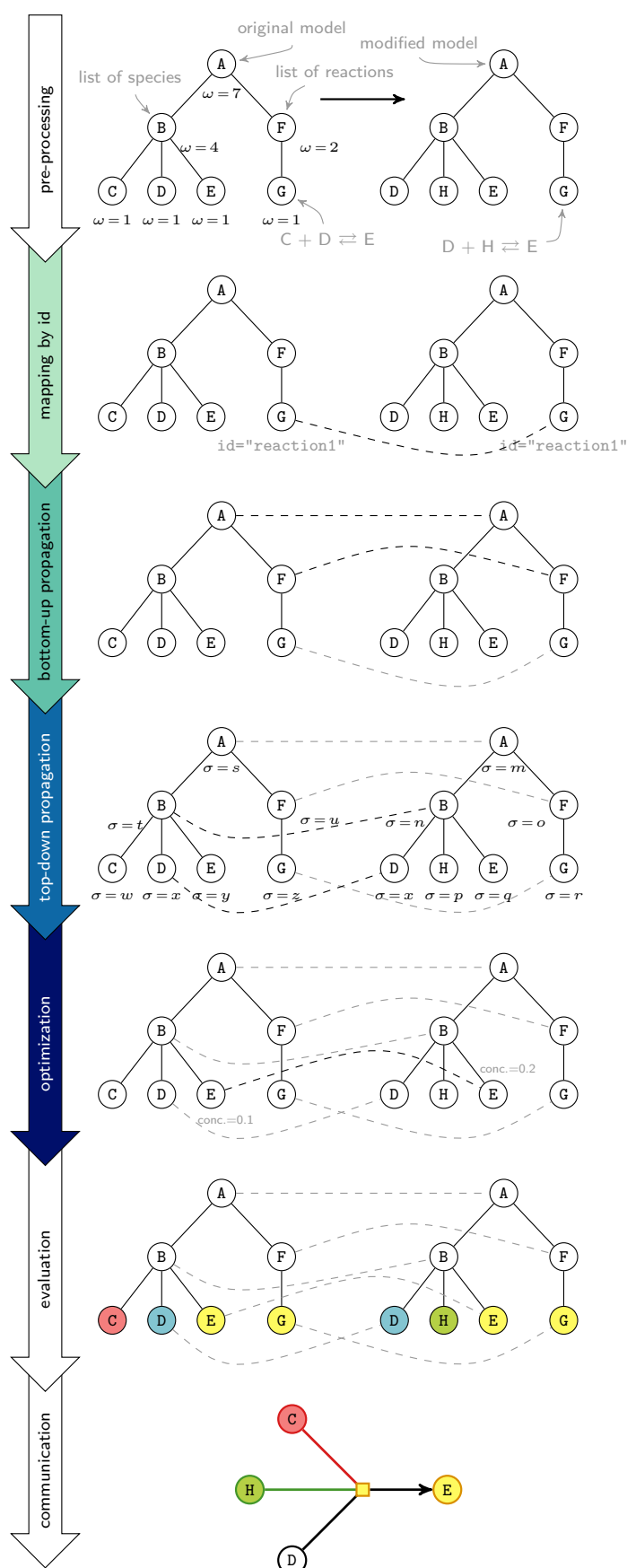


Figure 2.2. Schematic of the mapping procedure. The figure illustrates the procedure to communicate the differences between two versions of a model (row one) to the user (row seven). The models are presented as abstract trees, following the SBML structure. The trees on the left-hand side correspond to the original version of the model, the trees on the right-hand side illustrate the modified version of the model. Nodes A-H represent single entities in the model documents: A represents the model node; B represents the `listOfSpecies` node, hosting the species C, D, E, and H; F represents the `listOfReactions` node, hosting the reaction G. Dashed lines indicate mappings between nodes of the two document versions. The values of σ and ω represent signatures and weights of nodes. They are calculated during the pre-processing step, see Section 2.2.1. The different colours of the nodes indicate modifications: Updates are yellow, inserts are green, deletes are red. In the evaluation step, moves are blue.

of these children, a 2×2 distance matrix is created. The elements in this matrix represent differences between the attributes of the corresponding nodes. The E -nodes only differ in the value of the concentration attribute. Changing the value of one single attribute in a species is a minor update and, thus, the nodes' distance is very small. In contrast, the nodes C , E , and H do not have anything in common. Consequently, the E -nodes will be mapped while C and H remain unmapped. Finally, the resulting mapping is analysed (row six, *evaluation*). For example, the algorithm detects that D was deleted, H was inserted and E was modified in version 2. The difference graph, as obtained when interpreting the results of the *evaluation* step, is shown on the bottom of Figure 2.2. Particular means of communicating the differences are described in Figure 2.4.

2.3. Implementing the algorithm for detection and communication of differences

The devised algorithm is implemented in a software library, BiVeS (Section 2.3.1). The web-based prototype BudHat demonstrates the power of BiVeS (Section 2.3.2). The JavaScript library DiVil visualises the differences in a standardised format (Section 2.3.3).

2.3.1. BiVeS detects and communicates differences between model versions

I developed a software library called BiVeS, which implements the described algorithm to detect and communicate differences in computational models. BiVeS is implemented in Java⁹. As shown in Figure 2.3, it consists of multiple modules: (i) the `xmlutils`¹⁰ provide sophisticated methods to read and manipulate XML documents, (ii) `BiVeS-Core`¹¹ implements different mapping strategies, (iii) `BiVeS`¹² integrates all modules and provides a framework and an API to execute a model comparison, (iv) using the `jCOMODI`¹³ library BiVeS is able to automatically annotate differences, (v) `BiVeS-WebApp`¹⁴ implements a web interface to compare models remotely, and (vi) `BiVeS-WebApp-Client`¹⁵ can be used by other software projects to communicate with `BiVeS-WebApp`. In addition, the modules (vii) `BiVeS-SBML`¹⁶ and (viii) `BiVeS-CellML`¹⁷ make BiVeS understand the domain characteristics of models encoded in SBML and CellML, respectively. Thus, the results of a comparison of models in these formats are more meaningful and easier to comprehend. However, BiVeS is also able to handle arbitrary XML documents.

⁹java.com, accessed 20 May 2017

¹⁰semsproject.github.io/XMLUtils, accessed 31 May 2017

¹¹semsproject.github.io/BiVeS-Core, accessed 31 May 2017

¹²semsproject.github.io/BiVeS, accessed 31 May 2017

¹³semsproject.github.io/jCOMODI, accessed 31 May 2017

¹⁴semsproject.github.io/BiVeS-WS, accessed 31 May 2017. A demo instance including a tiny how-to can be found at bives.bio.informatik.uni-rostock.de

¹⁵semsproject.github.io/BiVeS-WS-Client, accessed 31 May 2017

¹⁶semsproject.github.io/BiVeS-SBML, accessed 31 May 2017

¹⁷semsproject.github.io/BiVeS-CellML, accessed 31 May 2017

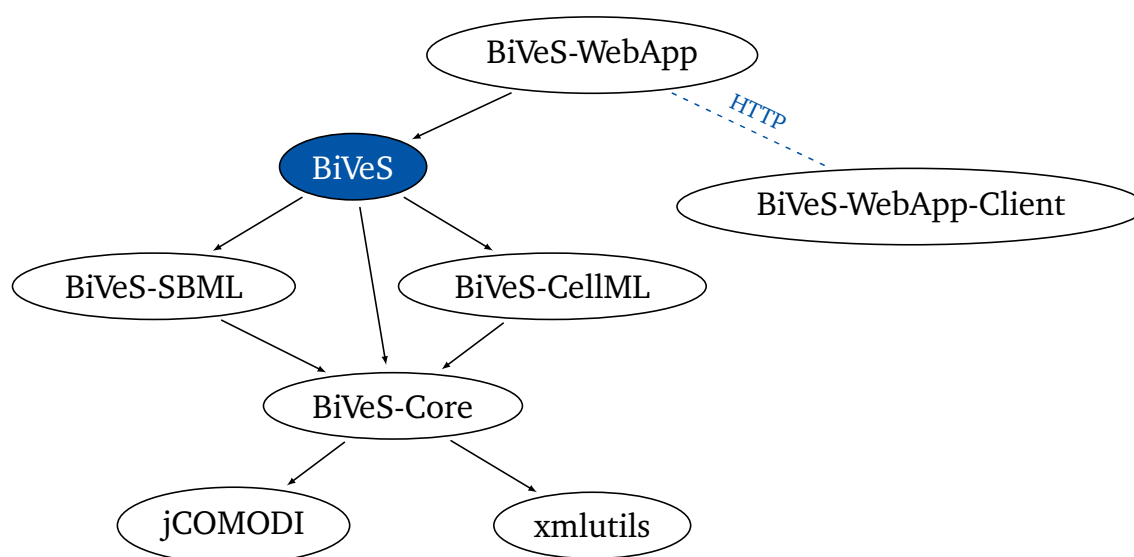


Figure 2.3. Dependency graph of BiVeS' modules. BiVeS consists of different modules. The main module providing full API-access and a command-line interface is highlighted in blue. The modules BiVeS-SBML and BiVeS-CellML implement format-specific domain characteristics and can optionally be incorporated at run time. There are more dependencies to software not directly related to this work, such as jDOM2, commons-cli, or gson. Moreover, as the modules use unit tests to validate the implementation all modules also depend on jUnit.

Further modules may be implemented in the future to also support other modelling languages without the need for reimplementing the core algorithms. Maven¹⁸ is used to handle the dependencies and to manage the life cycle of deployments.

BiVeS exports the difference graph in several output formats, including computer-digestible XML code and a graphical representation. One type of output are XML encoded, machine-readable deltas, which describe the differences between two versions of a model (see Section 2.2.4). A remarkable feature of these deltas is their completeness. They can be inverted and composed [Mar+01]. That means, given one model version and the delta, the opposite version can be retrieved [SO09].

Another major feature is the translation of the delta into human-readable formats (refer to the *communication* step in Figure 2.2). BiVeS summarises the model-related changes in a text-based report. This type of output is ideally suited to be integrated in other tools. Specifically, the report is either encoded in Markdown, ReStructuredText or HTML. The report in HTML format is generated for convenience, e. g., to instantly display the changes on a web page. Figure 2.4b shows a sample report. Another notable feature is the encoding of differences in standard graph representations enabling a subsequent visualisation. While BiVeS itself cannot produce rendered graphical output, it exports different graphical notations, including GraphML, Dot,

¹⁸maven.apache.org, accessed 15 June 2017

or JSON¹⁹. These graphs can then be further processed or rendered by end-user software. Figure 2.4a shows such a highlighted reaction network, as produced by BudHat.

BiVeS currently supports SBML and CellML, but it could also be extended towards other XML-based model exchange formats such as NeuroML [Gle+10] or PharmML [Swa+15]. Moreover, BiVeS could improve version control of simulation descriptions (e. g., differences between two simulation setups encoded in SED-ML. Armed with this, it is possible to produce visualisations, as implemented in BudHat (Section 2.3.2) and MoSt (Section 3.4).

2.3.2. BudHat demonstrates the advantages of BiVeS

As a proof of concept, I implemented the web-based interface BudHat²⁰, which uses BiVeS to compare versions of a computational model. BudHat contains a rudimentary user management and stores models in a database back-end. It calls BiVeS using its Java API to compare models and displays the obtained results in the web browser. The different visualisations generated by BudHat are shown in Figure 2.4. All figures show the difference between versions from June 2007 and November 2013 of model BIOMD0000000107 in BioModels Database, compare Figure 2.1. The highlighted reaction network in Figure 2.1a is based on BiVeS' GraphML export and was rendered using Cytoscape Web [Lop+10]. The differences can be recomputed and visualised online²¹.

2.3.3. DiVil visualises the differences in standard formats

The DiVil library²², developed in our department at the University of Rostock, provides standardised visualisations of model differences on the internet. BiVeS exclusively exports machine-readable graph formats highlighting the differences between model versions. DiVil is able to understand BiVeS' output and to visualise reaction networks in SBGN compliant graphs. Based on D3²³, JQuery²⁴, and FileSaver.js²⁵, the visualisations are interactive and can be exported in SBGN-ML format.

The DiVil library has already been integrated in the MoSt platform to visualise differences between versions of computational models from open repositories.

¹⁹json.org, accessed 31 May 2017

²⁰budhat.bio.informatik.uni-rostock.de, accessed 31 May 2017

²¹s.binfalse.de/novak93diff, accessed 24 October 2017 – please note that BudHat has been discontinued as there are many other applications demonstrating the power of BiVeS, see Chapter 3

²²github.com/SemsProject/DiVil, accessed 23 July 2017

²³d3js.org, accessed 23 July 2017

²⁴jquery.com, accessed 23 July 2017

²⁵github.com/eligrey/FileSaver.js, accessed 23 July 2017

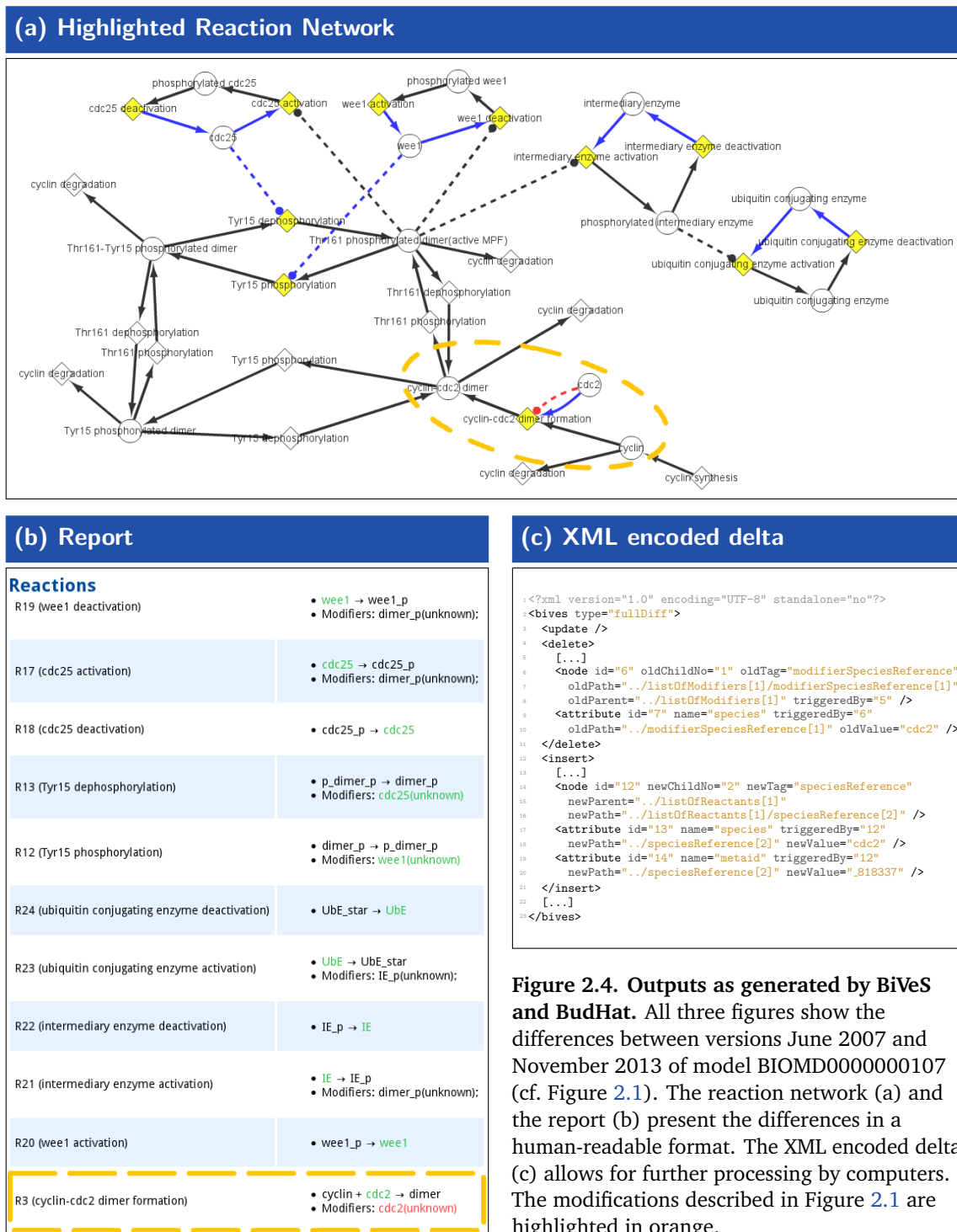


Figure 2.4. Outputs as generated by BiVeS and BudHat. All three figures show the differences between versions June 2007 and November 2013 of model BIOMD0000000107 (cf. Figure 2.1). The reaction network (a) and the report (b) present the differences in a human-readable format. The XML encoded delta (c) allows for further processing by computers. The modifications described in Figure 2.1 are highlighted in orange.

2.4. Semantic characterisation of differences

As discussed in Chapter 1, the multi-disciplinary approach in the life sciences requires scientists to reuse other works. Formats such as SBML and CellML have several advantages: Models can be simulated, analysed, and visualised using different software tools; models encoded in standard formats may outlive the tool used to create the model; model exchange becomes feasible; and models can more easily be shared, published, and reused [Sch+16]. Both formats focus on encoding the biological network, the mathematics, and the dynamics of the system. This information enables the technical reuse of model code. However, sustainable model reuse requires a basic understanding of (i) the biological background, (ii) the modelled system, and (iii) possible parametrisations under different conditions. Even though the communication channels presented in the previous sections are eminently useful for humans, machines still do not understand the differences and their effects. Using semantic annotations it is possible to encode knowledge about the differences in a machine-readable format. For this purpose, terms from bio-ontologies and controlled vocabularies can be linked to the model, adding a semantic layer. I propose a similar approach for the semantic description of differences between versions of a model.

Regular changes in models lead to different versions of a model [Wal+13]. For example, modellers test different hypotheses, maintainers of databases initially curate the model, and other scientists later on correct or extend it. Specifically, parameter values are updated, errors are corrected, models are adopted to changes in the underlying format, etc. On average, a model changes 4.69 times during the first five years after publishing in an open repository²⁶. It is important to track these changes for a number of reasons. For example, changes in parametrisations or on the underlying network may lead to a situation where the original results are not reproducible anymore. Furthermore, all contributions to the model code should be correctly attributed. With respect to simulation results, change records may help to predict modifications in the simulation outcome. Using the semantic layer to describe changes in a model allows for storing meaning together with possible implications of these changes. Changes can then be filtered and analysed automatically. Ultimately, a good communication of model changes increases the trust of scientists wishing to reuse a model for their own purposes.

2.4.1. The COMODI ontology

In the Section 2.2, I introduced an algorithm to identify and communicate the changes between two versions of a model. The corresponding software tool BiVeS encodes changes in an XML file and, thus, other tools can visualise and post-process identified changes. Small changes might be easy to grasp without the aid of a principled annotation scheme. However, as the list of changes increases it becomes harder to understand their relevance. To address this problem,

²⁶most.bio.informatik.uni-rostock.de, accessed 30 May 2017

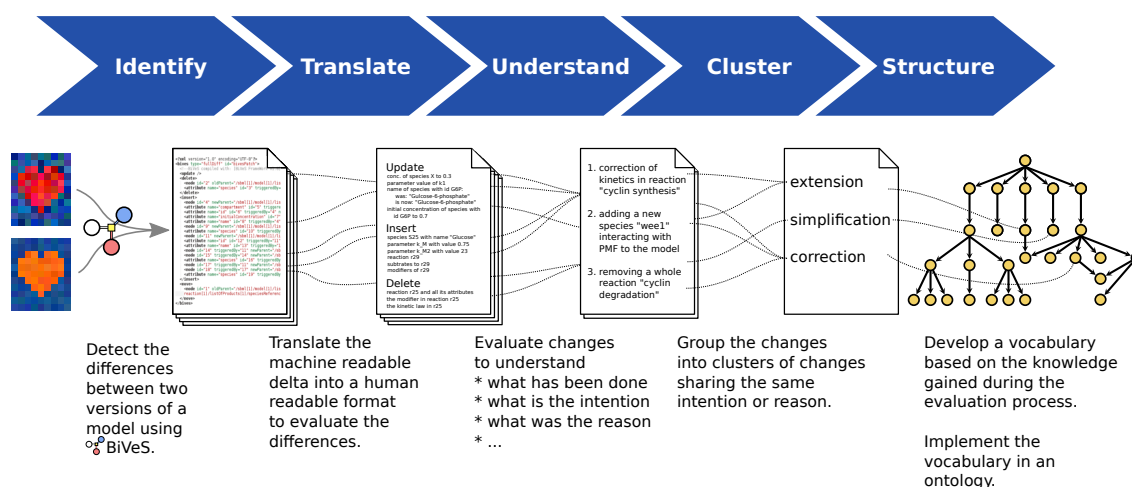


Figure 2.5. Development process of COMODI. The development process involved five steps with several iterations. First, I used BiVeS to compute the differences between all subsequent model versions. Second, I converted the formal description of more than 10 000 differences into human-readable descriptions. Third, I manually studied these descriptions and derived hypotheses and explanations for them. Fourth, I grouped the human-readable descriptions into sets of concepts and derived candidate terms for the ontology. Fifth, I aggregated and classified these terms and implemented the first version of the ontology in Protégé.

I developed an ontology of terms describing changes in models. The ontology can be used by scientists and within software to characterise model updates at the level of single changes.

In this section I present COMODI, an ontology needed because COMputational MODels Differ. It empowers users and software to describe changes in a model on the semantic level. When studying or reusing a model, such annotations help with determining the relevance of a change in a given context. COMODI also enables software to implement user-specific filter options for model changes. Finally, COMODI is a step towards predicting how a change in a model influences the simulation results. The ontology can be used to annotate differences between computational models, including those encoded in SBML and CellML.

Even though I had major contributions on the work described in the following, we conducted the study in a team including two students. Nevertheless, for stylistic reasons I am describing the work from a singular perspective and, thus, remain with the first-person *I* instead of a *we*. However, this does not lower the importance and value of my colleague's contributions.

2.4.2. Design considerations for the ontology

COMODI was developed based on a study of changes in versions of SBML and CellML models. The models were retrieved from the respective model repositories. More specifically, I started my investigation by manually analysing a predefined set of cell cycle models²⁷ from BioModels Database. I subsequently extended this set with randomly chosen models from both, BioModels

²⁷108 versions of the models with IDs [BIOMD0000000005](#), [BIOMD0000000006](#), [BIOMD0000000007](#), [BIOMD0000000056](#), and [BIOMD0000000107](#)

Chapter 2. A method to characterise differences in computational models

Database and the Physiome Model Repository. The single steps of development are summarised in Figure 2.5 and explained in the following:

1. Using the BiVeS algorithm I identified the differences between all subsequent versions of each model and exported the deltas in XML-encoded files. A delta is a set of operations on entities in XML documents (nodes or attributes, respectively) necessary to transform one document into another, compare Section 2.2.4.
2. Each found difference was manually translated into a human-readable description and recorded in a wiki software to share and discuss it with collaborators. In total, I investigated more than 10 000 differences.
3. Afterwards, I manually analysed the verbose descriptions of changes to understand their effects on the model and to derive hypotheses and explanations for a change. For example, the change of an entity name from *Gulcose* to *Glucose* renames a species and can be considered as the `Correction` of a `Typo` that effects an `EntityName`.
4. I then grouped the changes into several logical clusters, according to the derived hypotheses and explanations of a set of changes. These clusters are based on my own experiences and on feedback from domain experts. The knowledge I gained led to candidate terms for the ontology. I used the human-readable description as a basis for the term definitions.
5. In a last step, I designed a first version of the ontology from the obtained clusters. The ontology was afterwards extended with concepts stemming from standard formats (SBML and CellML terminology, e.g. `ParameterSetup`) and from the XML domain (e.g. `EntityIdentifier`).

I quickly identified technically driven properties of changes. For example, it is easy to determine the type of a change as BiVeS already distinguishes between *insertions*, *deletions*, *updates*, and *moves* of entities in XML documents. Moreover, it is always possible to specify the XML entity that is subject to a change. It was, however, more difficult to identify terms describing the reason, intention, or target of a change. The absence of appropriate terms led me to derive new terms based on the human-readable description of changes. The initial set of terms was then shaped in discussions with other researchers. Throughout the development of COMODI, I sought feedback from experts in the fields, e.g., through personal communication or poster presentations at conferences. Finally, I implemented the derived ontology in the Web Ontology Language (OWL) [Bec09] using Protégé [Noy+03].

2.4.3. Ontology organisation and content

COMODI is organised into four branches around the central concept `Change: XmlEntity`, `Intention`, `Reason`, and `Target` (cf. Figure 2.6; the full class hierarchy is described in Appendix E.1). In the following I use the change of a parameter in an imaginary SBML model as a running example:

Chapter 2. A method to characterise differences in computational models

I assume that the parameter changed from 0.5 in the old version to 0.8 in the new version of the SBML model.

The subtree rooted by the `Change` class can be used to specify the type of a change in more detail. Model entities may be inserted (`Insertion`), deleted (`Deletion`), updated (`Update`), or moved (`Move`). In my example, the modification of the parameter value from 0.5 to 0.8 corresponds to an update (`Update`) of an attribute value.

Many models are encoded in XML documents. In these cases, a change is always applied to a certain `XmlEntity`. I distinguish between an `XmlNode`, an `XmlAttribute`, or an `XmlText` element. The update of the parameter value in my example is applied to an `XmlAttribute`.

`Intention` and `Reason` both indicate the purpose of a change. On the one hand, the `Intention` specifies the aim of a change, particularly with respect to consequences in the future. In my example, the intention of modifying the parameter value is a `Correction`. On the other hand, a `Reason` specifically focuses on the cause of a change. In my example, a `MismatchWithPublication` caused an update of the parameter value.

Most prominent is the `Target` branch. It contains terms to specify possible targets of a change. COMODI basically distinguishes between five layers in a model document, that can be subject to a change:

1. The `ModelEncoding` corresponds to the formal encoding of the model document. Terms of this branch can, for example, be used to describe an update of the underlying SBML specification.
2. The `ModelAnnotation` branch corresponds to the semantic layer of a model document. Terms of this branch can, for example, be used to capture changes in the annotations.
3. The `ModelDefinition` refers to the actual biological system, for example a reaction network. Terms of this branch can, for example, be used to specify the parts of a model that are affected by a change.
4. The `ModelSetup` branch can be used to describe changes in the simulation environment. Terms of this branch can, for example, be used to describe changes in parameter values.
5. The `ModelBehaviour` links to the TEDDY ontology [Cou+11]. Thus, it is possible to capture changes in the dynamics of the system. Such changes may, for example, affect the stability characteristics.

My example affects the `ParameterSetup`, which belongs to the `ModelSetup`.

Finally, different changes might be linked if they have mutual dependencies. For example, the deletion of a biological reaction triggers the deletion of its kinetic law. Similarly, the deletion of an XML node (e.g. an SBML species) triggers the deletion of all its attributes (e.g. the species' initial concentration). Those changes can be linked using the `wasTriggeredBy` relationship to express relations between changes.

COMODI version 2017-10-11 contains a hierarchy of 65 classes and includes five object properties. The object properties can be used to establish relationships between members of

Chapter 2. A method to characterise differences in computational models

Name	Description	Domain	Range
affects	Provides information about the parts in a model that were affected by a change.	Change	Target
appliesTo	Stores information about the entity type in an XML document that was changed.	Change	XmlEntity
hasIntention	Links a change to an intention that was to be achieved by the corresponding change.	Change	Intention
hasReason	Links a change to a reason that made this change necessary.	Change	Reason
wasTriggeredBy	Represents dependencies among changes: A change might trigger further changes.	Change	Change

Table 2.1. List of object properties defined in COMODI.

the `Change` class and members of the four main branches of the ontology. I list and explain these properties in Table 2.1.

The COMODI ontology is specifically designed for the annotation of differences between versions of a computational model in the life sciences. In the following I show the usefulness of COMODI for annotating changes, predicting the effects of changes on the simulation result, and filtering versions of a model for specific differences.

2.4.4. Annotation of changes

SBML models typically use parameters to define the kinetics of a process. The corresponding entity in the SBML document may look as follows:

```
1 <parameter name="Km1" value="23.24" units="molesperlitre" />
```

Here the value of the parameter `Km1` is 23.24 molesperlitre. Updating the parameter value to 23.42 molesperlitre results in an update of the corresponding XML entity. The new version of the model then contains the following piece of SBML code:

```
1 <parameter name="Km1" value="23.42" units="molesperlitre" />
```

BiVeS identifies the difference as an update of the parameter value. The corresponding snippet in the XML-encoded serialisation provides the new and the old value of `Km1`:

```
1 <update>
2   <attribute id="1" oldPath="/sbml[1]/[...]/parameter[1]" name="value"
3     newValue="23.42" oldValue="23.24" [...] />
4 </update>
```

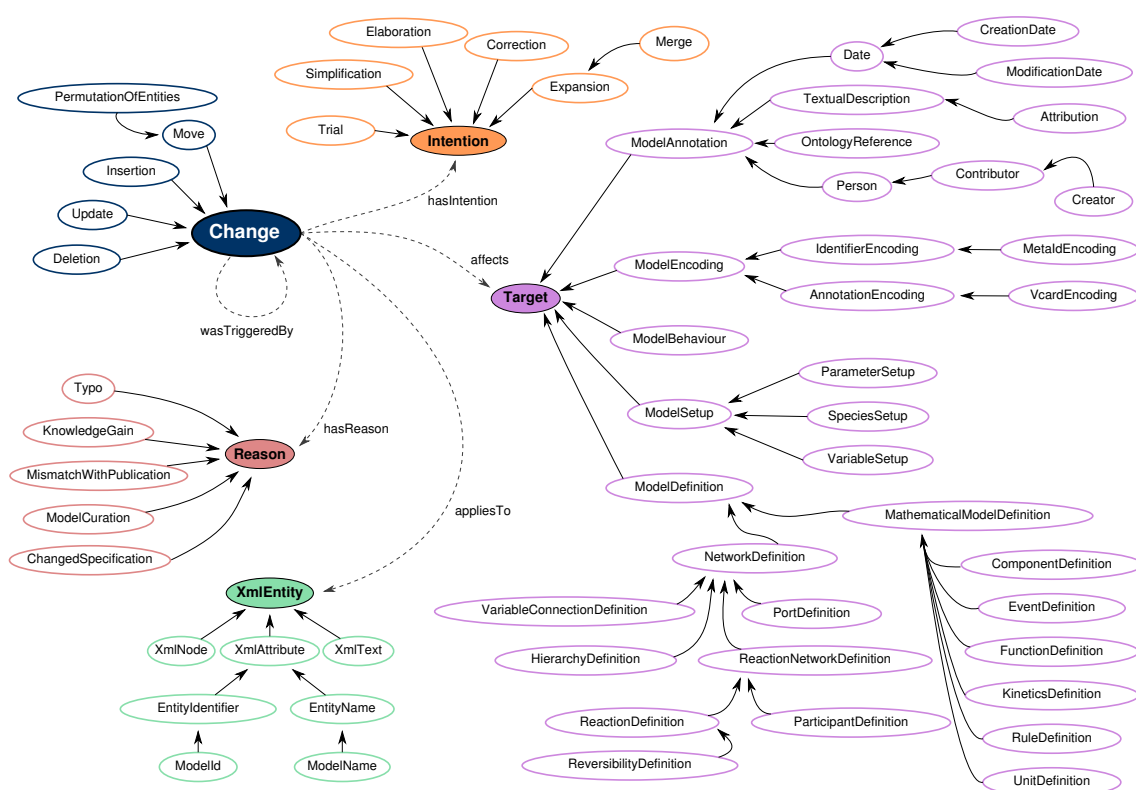



Figure 2.6. Structure of the COMODI ontology. Differences between computational models can be annotated with the *Change* term. Using the properties *appliesTo*, *hasIntention*, *hasReason*, and *affects*, the differences can be linked to the terms of the four major branches of COMODI: *XmlEntity*, *Intention*, *Reason* and *Target*. All arrows between terms within these five branches indicate an *is-a* relation, unless labelled otherwise.

Using COMODI the detected update can now be annotated. Some information can directly be inferred and thus be annotated automatically with BiVeS. For example, I know that the change is an update and can link it to the XML entity *XmlAttribute*. BiVeS is even able to recognise that this change corresponds to a change of the *ParameterSetup*. The combination of several statements using terms of the different branches allows users to be very specific. COMODI offers terms describing the reason and the intention of a change. Following the example from the previous section, the annotation of the parameter update is shown in Source Code 2.1.

2.4.5. Prediction of possible consequences of a change

The modification of the *ParameterSetup* also affects the *ModelSetup* (cf. ontology terms in Figure 2.6) and thus potentially influences the simulation results. Similarly, modifications of a *FunctionDefinition* or the *KineticsDefinition* can influence the simulation outcome. Finally, changes in the network structure (e.g., modification of the *ReactionNetworkDefinition* by transforming a reactant into a modifier) will not only affect the simulation outcome, but in addition

Chapter 2. A method to characterise differences in computational models

```
1 <#1> a comodi:Update ;  
2   comodi:appliesTo comodi:XmlAttribute ;  
3   comodi:affects comodi:ParameterSetup;  
4   comodi:hasIntention comodi:Correction ;  
5   comodi:hasReason comodi:MismatchWithPublication .
```

Source Code 2.1. Annotation of a difference using terms from the COMODI ontology. The code shows the annotation of difference #1 as an (1) update, that (2) applies to an attribute in the XML document, (3) affects the setup of a parameter in the model, (4) represents a correction of the model, and (5) was done because the model did not match the original publication. This annotation is serialised using the TURTLE format, see w3.org/TR/turtle.

the visual representation of the network. These changes are potentially relevant for modellers, so they should be notified of any modification.

In contrast, changes that affect the `ModelEncoding` may be irrelevant for modellers. For example, models are regularly updated to remain compliant with new versions of format specifications. These changes are, however, relevant for software tools dealing with model code. As not all tools feature the full set of SBML constructs [Huc+11] the upgrade of a model may require the use of another software tool. Thus, changes that result from modifications of the format specification can be of indirect interest for modellers. They may not affect the modelled system, but the tools that are needed to interpret and simulate it.

Other changes may be absolutely irrelevant. It can be helpful to hide them and thereby help users focus on important changes. For example, the reading and subsequent writing of a model file using different software tools, such as COPASI [Hoo+06] or CellDesigner [Fun+03], often results in a re-shuffling of elements within the document. However, the sequence of certain elements might not matter to the encoded model. In SBML for example, the order of parameters defined in the `listOfParameters` is irrelevant for the encoded system, as SBML does not give any semantic meaning to element orders [Huc+15a]. Thus, changes that only affect the order of parameters, can be neglected. Even if BiVeS reports them in its XML serialisation, these changes can be discarded if annotated with the corresponding COMODI terms. For other types of changes, the decision whether to neglect a change or not depends on the user. A new identifier scheme for the semantic annotations, for example, is relevant to curators and tool developers, while it is probably irrelevant for the majority of modellers. However, modellers who based their model analysis, comparison, or visualisation on semantic annotations need to be notified about this type of change. Here, COMODI terms need to be evaluated based on the users' preferences.

2.4.6. Filter for changes

The COMODI ontology enables software tools to automatically filter the list of changes and to show only relevant changes for a given question. For example, if a developer is interested in

Chapter 2. A method to characterise differences in computational models

the model versions before and after an update of the SBML specification, he or she can search for changes annotated with `ChangedSpecification` and study only those versions of a model that are linked to such a change.

Another, more complex filter is the one for “relevant changes only”. It is difficult to determine what exactly a relevant change may be, as relevance depends on the application domain and user. However, in the context of curation, the curators could define their set of changes that they want displayed and neglected, respectively. The needs of specific user groups may result in different filter-profiles.

Filtering can also be used to display only model versions that are the result of a specific change, while neglecting all other versions. For example, each release of BioModels Database generates new model versions. However, if the only changes are updates of the SBML level, then it suffices to display a reduced number of model versions to the user, instead of providing all released versions.

2.5. Conclusions and impact

Reproducibility of model-based scientific results has gained increasing attention [CF10; Gen05; Pen11; San+13]. Indeed, the ability to reproduce results is a basic requirement for the advance of science [Kin+11]. However, the reuse of models requires the accessibility and comparability of models and their versions. Model provenance and version control enable the widespread use and application of models, saving time and efforts during development [Mes10; WW16]. Model repositories have been working on this for the past decade and provide access to computational models described in scientific publications. Support for version control, however, is still limited. Existing implementations rely on standard version control systems and do not consider the specific requirements of modelling in the domain of computational biology [Wal+13]. Model repositories, such as BioModels Database and the Physiome Model Repository, can benefit from integrating the method, presented in my thesis, with their solutions for version control. Such a combined system stores model versions and detects the differences between them. In addition, it offers support for understanding and filtering changes according to the users’ preferences.

The presented tools improve difference detection for model versions. Standard formats describing computational models in biology are based on XML. Changes in versions of these models are typically computed with Unix’ diff, which performs badly on XML documents because it uses a line-based algorithm [Mye86]. BiVeS, on the other hand, is designed to respect the characteristics of domain-specific XML documents and to produce meaningful deltas. Its major advantages over existing solutions for biological models are: (i) It recognises the models’ hierarchical structures, (ii) it ignores white spaces, which do not affect the model, (iii) it ignores

Chapter 2. *A method to characterise differences in computational models*

the specific order of attributes in an entity. Additional post-processing rules capture the domain characteristics of the processed data and increase the significance of produced deltas.

BiVeS produces reports and graphical representations of differences using open formats to communicate the changes. Together with BudHat and DiVil these representations can be visualised, e.g., in standard SBGN format. Figure 2.4a shows how the visualisation supports users in exploring the changes affecting the biological network. Additionally, BiVeS compiles a comprehensive list of changes into a human-readable report, as shown in Figure 2.4b. Reports are particularly suitable for people interested in the details of mathematical changes. BiVeS' outputs can of course be used by other tools for further processing of results, compare Figure 2.4c.

In summary, the presented tools improve the detection of differences between versions of models in SBML or CellML format. All tools are released as open source software. The tools are available both as source code through, e.g., GitHub²⁸, and as binaries from our servers²⁹. There are multiple ways to integrate the tools in other software tools, as presented in Chapter 3. Everyone is invited to contribute code, comments, and feature requests.

COMODI helps capturing a model's evolution. COMODI is an ontology to describe the differences between versions of a computational model. The ontology terms specify the type of change for each detected difference. Usually, a combination of COMODI terms from different branches is necessary to characterise a change sufficiently. COMODI is currently used for the description of changes between two versions of the same model, either encoded in SBML or in CellML. Furthermore, COMODI terms can be applied to differences detected between models in any other encoding format, including even code from proprietary languages such as MATLAB³⁰.

I developed the COMODI ontology based on a manual study of changes in versions of curated SBML models from BioModels Database and in versions of CellML models from the Physiome Model Repository. These models, however, were all implemented in core SBML and core CellML. That means, I did not consider extensions. In the future, the study should be extended to also cover SBML models that use SBML extension packages.

The COMODI ontology encodes knowledge on the model level. Already now, tools such as SEEK, JWS, or COPASI can benefit from storing and evaluating information about model changes using COMODI. The ontology is also useful for recording the history of a model, and it ensures better transparency of a model's evolution. Furthermore, it enhances the traceability of updates and error corrections in existing models. COMODI cannot, however, be used to encode provenance, such as information about the user who changed the model or information about the tool used to update the file. It can, however, easily be coupled with ontologies for provenance. Specifically, PROV [MG13] and PAV [Cic+13] offer some compelling concepts

²⁸github.com/SemsProject, accessed 4 June 2017

²⁹bin.bio.informatik.uni-rostock.de, accessed 4 June 2017

³⁰www.mathworks.com/products/matlab.html, accessed 24 October 2017

for model provenance, which modelling tools and platforms should take the responsibility of implementing support for. For example, the developers of COPASI are currently implementing mechanisms to allow users to easily keep a record of model versions. Each version will be documented by the modeller with free text comments but these are in natural language and therefore not easily machine-readable. To allow for machine-readable annotations, the software will also facilitate users to specify COMODI terms as version annotations. Additionally to tracking versions, which are user-definable, COPASI will also track a full provenance log for each model; this is a complete history of the model changes recorded automatically as they happen and serialised in a machine-readable XML format. COMODI terms will then be particularly useful to annotate all the changes as they happen.

COMODI is encoded in OWL. It is openly available at comodi.bio.informatik.uni-rostock.de. The COMODI ontology is licensed under the terms of the Creative Commons Attribution-ShareAlike 4.0 International License³¹. The OWL encoding of the latest version may be downloaded from purl.uni-rostock.de/comodi/comodi.owl. Additionally, users may browse the ontology at purl.uni-rostock.de/comodi/. I also registered COMODI at BioPortal [Whe+11]. It is available at bioportal.bioontology.org/ontologies/COMODI.

The presented method helps grasping the evolution of computational models. My novel algorithm identifies structures in the XML trees that model versions have in common and detects the differences. COMODI, coupled with my algorithm for difference detection, ensures the transparency of a model's evolution, and it enhances the traceability of updates and error corrections. Consequently, existing model repositories can benefit from extending their software and functionalities with version control. Gaining insights into the process of development of a particular model has the potential to increase the confidence in this model and supports the collaboration of distinct research projects dramatically. For this reason, difference detection plays a key role in model version control. Furthermore, provenance investigates the nature of differences in model versions, seeking answers to the seven W-questions: Who, What, Where, Why, When, Which, With (How)? [Gob02; Mor+08]. BiVeS contributes to the “What” and “How” as defined in [RL10]. The “What” refers to content related events, such as modifications of parameter values in the model, and non-content related events, such as the upgrade to a new SBML version. In addition, BiVeS tells you “How” the “What” has changed. Using COMODI it is possible to go into more details and to also address the “Why”.

In the following chapter I show how my method is applied in systems biology projects. I demonstrate how other tools integrate my work. Furthermore, I present the results of an analysis of the evolution of computational models in open repositories.

³¹creativecommons.org/licenses/by-sa/4.0, accessed 4 June 2017

CHAPTER 3

APPLYING THE METHOD FOR DIFFERENCE DETECTION IN SYSTEMS BIOLOGY PROJECTS

The method I introduced in the previous chapter already shows impact in the systems biology landscape. Several tools and databases, including SEEK and the Physiome Model Repository, use BiVeS to allow for comparison of versions of models. In combination, the algorithm and the ontology are able to filter identified differences and to drop all but biologically and mathematically relevant modifications. Thus, they support modellers in managing their models and in understanding changes between versions.

In this chapter I show how my method for difference detection can be applied and how ongoing projects use my implementation. The chapter is based on two refereed publications and a student's bachelor project. First, I demonstrate how popular systems biology projects use BiVeS with a special focus on the Cardiac Electrophysiology Web Lab, which I published in the Biophysical Journal [[CSM16](#)]. Second, a bachelor's thesis that I supervised extends an existing database scheme to integrate versions of models and detailed information on their differences [[Pet16](#)]. Third, I analyse the evolution of published studies in the two major repositories for systems biology models. I published this large scale study in BMC Systems Biology [[Sch+18](#)].

3.1. BiVeS in the wild

The algorithm introduced in Section 2.2 is implemented in the modular BiVeS library, compare Section 2.3. BiVeS reads models from plain XML strings or from XML documents located in a locally accessible file system. Moreover, BiVeS is also able to handle links to remote files. It downloads the files from given URLs automatically to a temporary directory for further processing. The XML documents are then parsed and converted into an internal object structure. These objects are then (i) pre-processed, (ii) mapped, (iii) post-processed, and (iv) evaluated according to the presented algorithm. Identified changes can be annotated semantically using terms from the COMODI ontology, as explained in Section 2.4. The resulting mapping and identified differences can be exported in several human and machine-readable formats, including an XML encoded patch, an HTML encoded report, or various graph formats. BiVeS was designed for an easy integration with other software tools. It can be used in three different ways:

BiVeS provides a smart API for comparison of model versions. Thus, other Java-based tools can just integrate BiVeS as a library and get full support for the comparison strategies. Calling BiVeS to compare two models is as easy as copying a few lines of code, as shown in Source Code 3.1. Through that API it is also possible to change the mapping strategies or to neglect or penalise certain mismatches. Detected differences can then be obtained in various formats, as described earlier. From Java, internally created objects can be reused and processed subsequently using custom algorithms. This API is, for instance, used by my open source prototype BudHat, the Masymos extension (see Section 3.3), and the StatsGenerator (see Section 3.4) providing plenty of example code.

```
1 URL v1 = new URL ("https://s.binfalse.de/diss-v1");
2 URL v2 = new URL ("https://s.binfalse.de/diss-v2");
3
4 TreeDocument td1 = new TreeDocument (XmlTools.readDocument (v1), v1.toURI ());
5 TreeDocument td2 = new TreeDocument (XmlTools.readDocument (v2), v2.toURI ());
6
7 Diff diff = new SBMLDiff (td1, td2);
8 diff.mapTrees ();
9 System.out.println (diff.getHTMLReport ());
```

Source Code 3.1. Calling BiVeS through its Java API. The first two lines define the locations of the document versions to be compared. Lines 4 and 5 make the library retrieve and parse both XML documents into an internal structure. The last three lines finally execute the comparison job and print the resulting HTML report on the command line. Please note that the server at `s.binfalse.de` is using *Let's Encrypt* (letsencrypt.org). Thus, if you are running Java prior to version 8u101 you need to install *IdenTrust* CA manually, see [letsencrypt.org/certificates s.binfalse.de/jtrust-le](https://letsencrypt.org/certificates/s.binfalse.de/jtrust-le) and s.binfalse.de/j8u101.

```
1 usr@srv $ curl -d '{
2   "files":
3   [
4     "https://s.binfalse.de/diss-v1",
5     "https://s.binfalse.de/diss-v2"
6   ],
7   "commands":
8   [
9     "SBML",
10    "reportHtml"
11  ]
12 }' https://bives.bio.informatik.uni-rostock.de
```

Source Code 3.2. Comparing models using the BiVeS web application. A JSON object encoding the comparison job of a SBML model in two versions (<https://s.binfalse.de/diss-v1> and <https://s.binfalse.de/diss-v2>) is sent via HTTP POST to the BiVeS web application running at <https://bives.bio.informatik.uni-rostock.de>. Here, the request asks for the HTML report of differences. The result will also be serialised in JSON. Thus, it is possible to request different actions in just one HTTP POST call. The code basically reassembles the API call shown in Source Code 3.1.

BiVeS is available as a web application to facilitate the integration with non-Java applications. Thus, it is possible to outsource comparison jobs to remote servers with just a few basic requirements for applications: They need to (i) have network access, (ii) speak HTTP, and (iii) understand the JSON format. Using Maven¹ the BiVeS web application compiles to a web application archive (WAR). This WAR package can be installed on Java-based web servers, such as Apache Tomcat², to register a new instance of the BiVeS web application. Model documents can be specified by uploading plain XML code or by submitting links to files on a remote machine. Three types of commands are available to specify a concrete task for BiVeS and to request the desired output. First, comparison commands expect two files submitted with a request and, optionally, the desired output, such as the XML patch (default) or the HTML report. An example of a comparison job submitted to the BiVeS web application is shown in Source Code 3.2. Second, additional commands help analysing model files. They just expect a single file and, for example, (i) flatten a hierarchical model, (ii) extract its reaction network, or (iii) return some metadata about the model. Source Code 3.3 shows how information about a model document can be obtained from a command line. Third, general commands may force BiVeS to treat a document as either SBML, CellML, or regular XML skipping any model related optimisation.

Client and server communicate through JSON over the HTTP protocol. The BiVeS web application is also available as a Docker image³ easing portability and deployment. Thus, everyone may run a private instance of the BiVeS web application. This, in turn, improves data

¹maven.apache.org, accessed 15 June 2017

²tomcat.apache.org, accessed 4 June 2017

³hub.docker.com/r/binfalse/bives-webapp, accessed 23 July 2017


```
1 usr@srv $ curl -sd '{
2   "files": [ "https://s.binfalse.de/diss-v1" ],
3   "commands": [
4     "documentType",
5     "meta"
6   ]
7 }' https://bives.bio.informatik.uni-rostock.de | python -mjson.tool
8 {
9   "documentType": [
10     "XML",
11     "SBML"
12   ],
13   "meta": {
14     "modelId": null,
15     "modelName": "test_model",
16     "nodestats": {
17       "compartment": 1,
18       "listOfCompartments": 1,
19       "listOfProducts": 1,
20       "listOfReactants": 1,
21       "listOfReactions": 1,
22       "listOfSpecies": 1,
23       "model": 1,
24       "reaction": 1,
25       "sbml": 1,
26       "species": 2,
27       "speciesReference": 2
28     },
29     "sbmlLevel": 2,
30     "sbmlVersion": 3
31   }
32 }
```

Source Code 3.3. Analysing model documents using the BiVeS web application. The command line requests some information about the model stored at <https://s.binfalse.de/diss-v1>. The output, beginning from line 8, shows, that the model is encoded in SBML Level 2 Version 3 and that it, for example, contains one reaction and two species.

privacy, protection, and security, as the model files do not need to be submitted to public web services. The Functional Curation⁴ project of Chaste [CSM16], for example, uses the BiVeS web service to track the evolution of models uploaded to their system, see Sections 3.2.3 and 4.6.

The BiVeS framework implements a main class and, therefore, it can be executed on a command line. Most of the above mentioned tasks are available through command line

⁴chaste.cs.ox.ac.uk/FunctionalCuration, accessed 4 June 2017

```
1 usr@srv $ java -jar BiVeS-1.11.1-jar-with-dependencies.jar --SBML
   ↳ https://s.binfalse.de/diss-v1 https://s.binfalse.de/diss-v2
2 <?xml version="1.0" encoding="UTF-8"?>
3 <bives type="fullDiff" id="bivesPatch">
4   <update>
5     <attribute name="initialConcentration" id="1" oldValue="100" newValue="120"
   ↳ oldPath="/sbml[1]/model[1]/listOfSpecies[1]/species[1]"
   ↳ newPath="/sbml[1]/model[1]/listOfSpecies[1]/species[1]" />
6   </update>
7   <insert>
8     <node id="2" newParent="/sbml[1]/model[1]/listOfSpecies[1]" newChildNo="3"
   ↳ newPath="/sbml[1]/model[1]/listOfSpecies[1]/species[3]" newTag="species" />
9     <!-- ... -->
10  </insert>
11 </bives>
```

Source Code 3.4. Calling BiVeS from the command line. The command line is shown in black on line 1. The Java archive (JAR) `BiVeS-1.11.1-jar-with-dependencies.jar` can, for example, be built using Maven or obtained as a pre-compiled binary from our web server at bin.bio.informatik.uni-rostock.de. The flag `--SBML` tells BiVeS to treat the documents as models encoded in SBML. The last two arguments point to the files that BiVeS should compare. The output, beginning from line 2, indicates the produced XML patch.

arguments and can, thus, be executed by calling the main class. A typical command line call and its output is shown in Figure 3.4. This procedure is especially useful for modellers whose repositories do not yet support versions of models. Terrifying, but true: Many repositories and management systems for computational models still do not provide proper version control or change detection. The command line interface empowers researchers to do the difference detection on their own. Furthermore, the command line option may also be useful for non-Java software tools that do not want to rely on the network (which is mandatory for the BiVeS web application). The data management platform SEEK [Wol+15], for example, implemented support for model version control by executing BiVeS as a sub-process, see Section 3.2.1.

The web site at sems.bio.informatik.uni-rostock.de/projects/bives offers further information about the three implementations, including examples, how-tos, the source code, and binaries of my tools.

3.2. Implementations of my method

The following sections demonstrate how popular projects in the systems biology domain implement and use the method for characterising differences in computational models. These sections include screenshots of web pages to indicate how identified differences are communicated to the users.

3.2.1. Versioning in SEEK and the FAIRDOM Hub

The FAIRDOM Hub is a web-based management system specialised for the heterogeneous nature of data in the life sciences [Wol+17]. It is a repository for data, standard operating procedures (SOP), and models. The data is organised according to the ISA infrastructure [San+12], a standardised format for describing how individual experiments are aggregated into wider studies and investigations. The hierarchical ISA data model⁵ distinguishes three main components: (i) Investigations are high level descriptions of the research in a project, (ii) Studies are units of research to answer a particular (biological) question, and (iii) Assays are particular experiments, measurements, or models. The platform implements a sophisticated permission system and provides services to integrate, interlink and publish all kinds of heterogeneous research results, rendering projects in the systems biology domain *findable*, *accessible*, *interoperable*, and *reusable* (FAIR). It bases on the SEEK platform and was developed in an European effort [Wol+11; Wol+15]. SEEK and, thus, the FAIRDOM Hub offer a RESTful⁶ API to access the data stored on the platform.

The FAIRDOM Hub allows data sharing through the full lifecycle of a project, and across projects. It implements a versioning concept, which tracks the history of assets. Thus, users can change their data and upload new versions. Versions carry numeric identifiers and the platform records metadata, such as the creator, the time stamps of its modification, and an optional revision comment. Models that are encoded in standard format can be compared using the BiVeS tool. An example is shown in Figure 3.1. The differences are colour coded, which makes it easy to spot the changes. On top of the comparison page BiVeS' HTML report is shown. It summarises the changes in five entities:

- The value of parameter k_B was modified.
- Parameter v_E , species E , and reaction s were inserted in the new version.
- Reaction r was modified. Species B is not a product anymore, but species E was added. In addition, the kinetic law of r was updated. The colour coded differences in the changed kinetic law supports the visitor in spotting the changes.

SEEK is open source, every project can install its own instance. As it is available as a Docker image the setup boils down to just a single command line call⁷. Nevertheless, an impressive number of users contribute model related data to the public FAIRDOM Hub instance. More than 800 people from almost 200 institutions are registered at fairdomhub.org. The web page lists 188 Models, 241 SOPs, and 1536 data files organised in 425 assays, 229 studies, and 117 investigations of 80 projects⁸.

⁵isa-tools.org, accessed 17 June 2017

⁶en.wikipedia.org/wiki/Representational_state_transfer, accessed 7 February 2018

⁷SEEK installation instructions at s.binfal.de/seek-docker, accessed 17 June 2017

⁸Not all data are publicly visible. Figures retrieved from fairdomhub.org, 17 June 2017

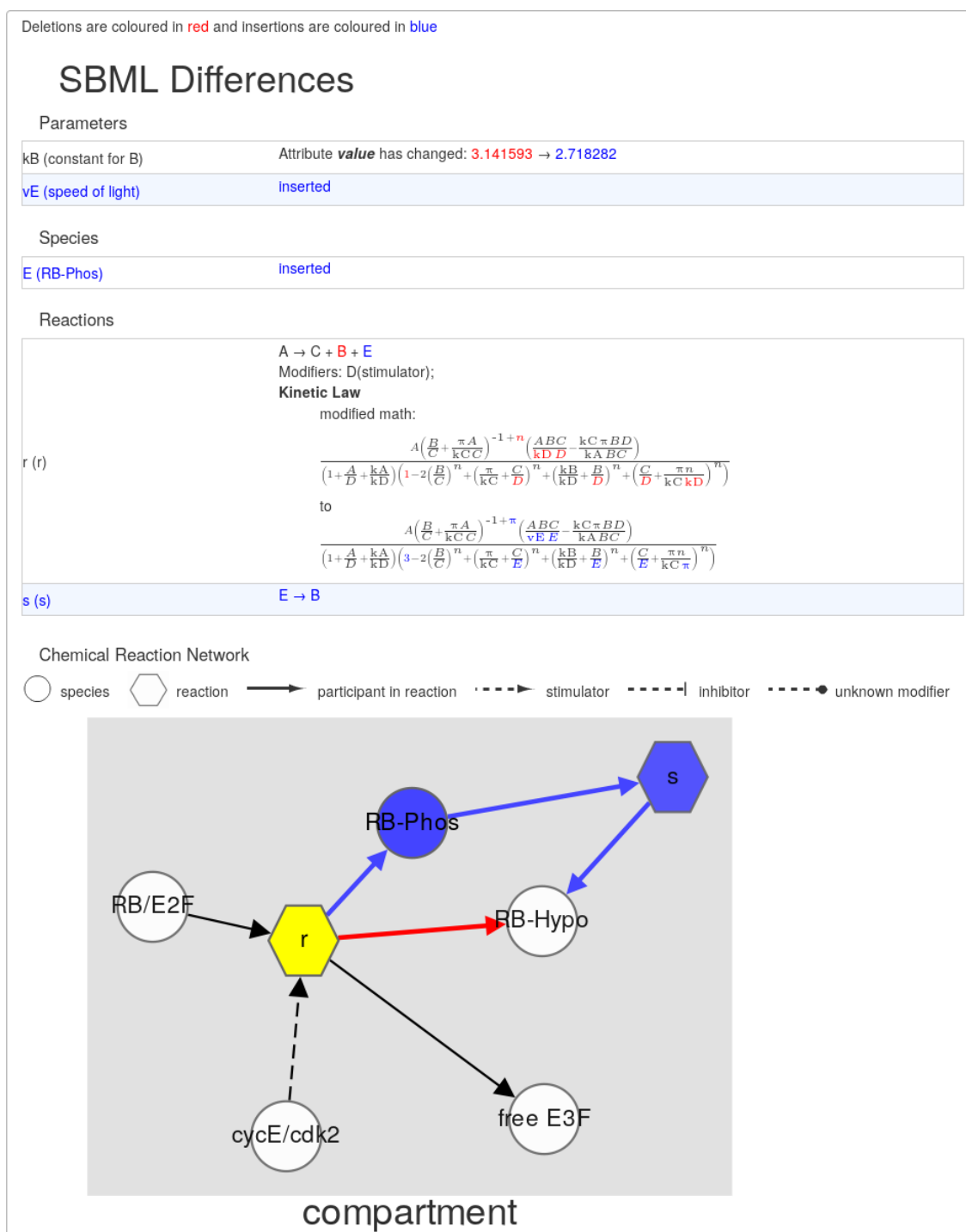


Figure 3.1. Screenshot of a model comparison result in the FAIRDOM Hub. The FAIRDOM Hub is based on SEEK, which integrates BiVeS for comparison of models in standard format. The comparison page shows two outputs of the BiVeS tool: The HTML report on top and the highlighted reaction network on the bottom. Inserts are coloured blue, deletes are red, and modifications are yellow. The comparison is also available online at s.binfalse.de/seek-demo.

3.2.2. Models and versions in the Physiome Model Repository

The Physiome Model Repository is a central resource for free-access computational models, primarily in CellML format [Yu+11]. Models are developed in so-called workspaces, which are sovereign partitions of the repository and typically represent distinct modelling projects. Every workspace is version controlled. The system records each change made to a model together with metadata, such as a time stamp, a change comment, and the author. Thus, the Physiome Model Repository encourages collaborative modelling. The developers of the Physiome Model Repository already understood the importance of version control, because “collaboration on a model can be greatly simplified by a tool which records the change history of a model, and makes that history available to other collaborators” [Yu+11]. The versioning in the back-end is done through Git⁹. Therefore, workspaces can be cloned and developed decentralised from everywhere. As the Git tool supports submodules it is possible to include workspaces into workspaces in the Physiome Model Repository, which harmonises perfectly with modularised CellML models: Modules and structures can be outsourced into specialised Git repositories [Cue+02]. Thus, the system promotes a modular model development. However, the latest version of a workspace is not necessarily the best version of the modelling project [Mil+11]. Therefore, particularly interesting and well documented revisions of workspaces can be published as so-called exposures in the Physiome Model Repository. Models and workspaces can be downloaded through the website or cloned using Git. As the Physiome Model Repository also integrates the CombineArchiveWeb application, workspaces can also be exported and shared with others using COMBINE archives, see Section 4.4.2.

The Git system used by the Physiome Model Repository already does difference detection for different versions of a file. However, by default it uses Unix’ diff utility and, thus, the comparison of versions is done line-based (the problem is outlined in Appendix A). Therefore, the Physiome Model Repository integrated the BiVeS tool for comparison of computational models. An example is shown in Figure 3.2. Similar to the FAIRDOM Hub, the differences are highlighted with colours, which makes it easy to spot the changes. On top of the comparison page the highlighted module hierarchy of the CellML model versions is shown. Below, the page shows BiVeS’ HTML report. The comparison in the example is much more complex, compared to the example above. However, it is still easy to see which entities were inserted, deleted, or have changed. For example, two new units were introduced (flux and millimolar), many of the variables in the UbE component have changed, but even the changes in the mathematical layer are easy to spot.

The Physiome Model Repository is available at models.physiomeproject.org/cellml and currently lists 701 workspaces and 595 exposures in 21 categories¹⁰.

⁹git-scm.com, accessed 18 June 2017

¹⁰Figures retrieved from models.physiomeproject.org, 18 June 2017

Chapter 3. Applying the method for difference detection in systems biology projects

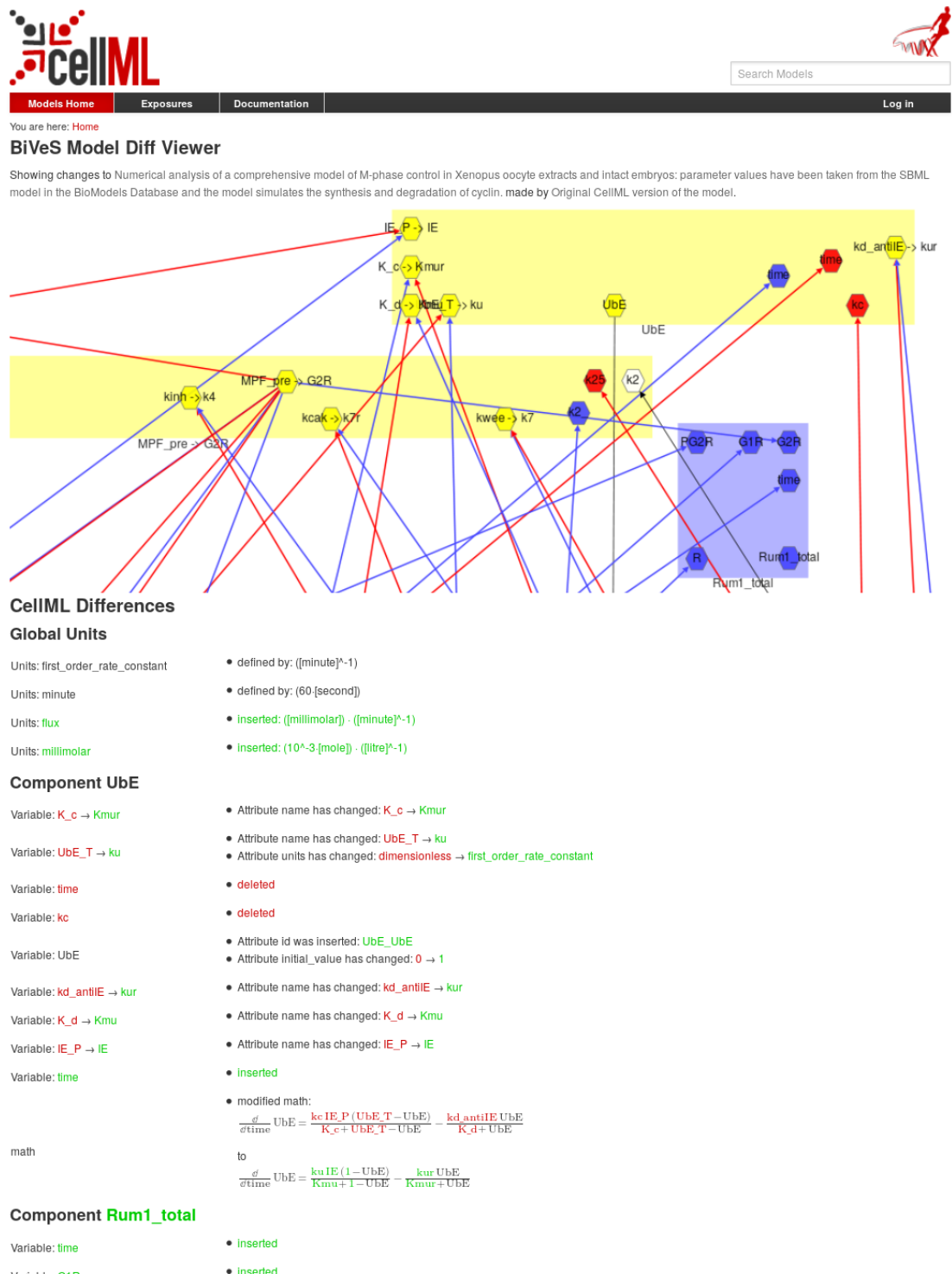


Figure 3.2. Screenshot of a model comparison result in the Physiome Model Repository. The comparison page shows two outputs of the BiVeS tool: The highlighted reaction network on the top and the HTML report on the bottom. Inserts are coloured green (in the HTML report) or blue (in the reaction network), deletes are red, and modifications are yellow. The screenshot shows the comparison results of the models s.binfal.se/pmr-novak93 and s.binfal.se/pmr-novak97.

3.2.3. Comparing models in the Cardiac Electrophysiology Web Lab

The fundamental concept of the Cardiac Electrophysiology Web Lab grounds on a decomposition of models and simulation descriptions. While simulation setups are traditionally merged into models, I encourage to distinguish between the definition of a biological system and its simulation setup [CSM16; CVW15]. The connection between a model and its setup should instead be established through semantic annotations, i.e., a simulation description may set the initial concentration of a parameter annotated with *is glucose*. Consequently, it will be easy to substitute a simulation description and analyse the model in a different environment and, vice versa, to evaluate or benchmark a number of models in a fixed environment.

The Cardiac Electrophysiology Web Lab was built with version control in mind. Models, protocols, and simulation results carry identifiers and are unambiguously accessible from the internet. For example, the *Decker 2009* model [Dec+09] exists in seven versions¹¹. Different versions of a model can be compared using the BiVeS tool. On the comparison page, the user can select versions as either predecessor or successor, difference detection is possible in both directions of time. A JavaScript framework sends the identifiers of selected versions to the web server at the University of Oxford, which then submits the comparison job to the BiVeS web application running at the University of Rostock. The differences detected by BiVeS are available as HTML report and as XML patch, both highlighted using colours, as shown in Figure 3.3. For example, the comparison of the model versions *Decker 2009 - buggy* and *Decker 2009 - fixed* lists changes in 28 CellML components¹². Parts of the changes on the component *CA* are shown in Figure 3.3. In the figure, one variable was deleted (*bss*), the initial concentrations in two variables have changed (*Ca_NSR* and *Ca_JSR*), and one variable was added to the model (*bss_cal*). In addition, the mathematical laws in that component has changed substantially. However, the colourisation makes it easy to spot the differences, even though the sequence of formulas has changed as well.

The Cardiac Electrophysiology Web Lab currently supports only cardiac models encoded in the CellML format and protocols in a proprietary format (see Section 4.6). At the moment of writing, the web lab contains 38 publicly available models and 25 public protocols. Moreover, models and protocols typically exist in multiple versions. The combination of a model version and a protocol version form a virtual experiment. However, not every model is compatible to every protocol. Results of successful virtual experiments can be compared visually with other experiments using modern techniques in the web browser. The Cardiac Electrophysiology Web Lab will also be focus of Section 4.6 in the next chapter.

¹¹All versions of the Decker 2009 model s.binfalse.de/decker-versions, accessed 30 June 2017

¹²Rerun the comparison online s.binfalse.de/decker-diff, accessed 1 July 2017

Comparison of Models

decker_2009.cellml

Available versions	Select as predecessor	Select as successor
Decker 2009 - fixed	<input type="radio"/>	<input checked="" type="radio"/>
Decker 2009 - buggy	<input checked="" type="radio"/>	<input type="radio"/>

Component Ca

Variable: bss

- deleted

Variable: Ca_NSR

- Attribute **initial_value** has changed: 0.929835335 → 1.01475649943057

Variable: Ca_JSR

- Attribute **initial_value** has changed: 0.917692717 → 0.993914988616979

Variable: bss_cal

- inserted

- modified math:

$$\frac{d}{dt} Ca_i = bmyo \left(\frac{-(ICab + IpCa - 2 INaCa) AF}{Vmyo \cdot 2} + \frac{(Ileak - Iup) Vnsr}{Vmyo} + \frac{Idiff Vss_CaL}{Vmyo} \right)$$

$$bmyo = \frac{1}{1 + \frac{cmdn_backm_cmdn}{(Ca_i + km_cmdn)^2} + \frac{km_trptnpu_bar}{(Ca_i + km_trpn)^2}}$$

$$\frac{d}{dt} Ca_ss_sr = bss_sr \left(- \left(Idiff + Idiff_ss - \left(\frac{2 INaCa_ss_sr AF}{2 Vss_sr} + \frac{Irel Vjsr}{Vss_sr} \right) \right) \right)$$

$$bss_sr = \frac{1}{1 + \frac{BSRmax \cdot KmBSR}{(KmBSR + Ca_ss_sr)^2} + \frac{BSLmax \cdot KmBSL}{(KmBSL + Ca_ss_sr)^2}}$$

$$\frac{d}{dt} Ca_ss_CaL = bss \left(- \left(\frac{ICaL AF}{2 Vss_CaL} - \frac{Idiff_ss Vss_sr}{Vss_CaL} \right) \right)$$

$$bss = \frac{1}{1 + \frac{BSRmax \cdot KmBSR}{(KmBSR + Ca_ss_CaL)^2} + \frac{BSLmax \cdot KmBSL}{(KmBSL + Ca_ss_CaL)^2}}$$

$$\frac{d}{dt} Ca_NSR = Iup - \left(Ileak + \frac{Itr Vjsr}{Vnsr} \right)$$

$$\frac{d}{dt} Ca_JSR = bcsqn (Itr - Irel)$$

$$bcsqn = \frac{1}{1 + \frac{kmcsqn \cdot csqphar}{(Ca_JSR + kmcsqn)^2}}$$

math

to

$$bmyo = \frac{1}{1 + \frac{cmdn_backm_cmdn}{(Ca_i + km_cmdn)^2} + \frac{km_trptnpu_bar}{(Ca_i + km_trpn)^2}}$$

$$\frac{d}{dt} Ca_i = bmyo \left(\frac{-(ICab + IpCa - 2 INaCa) AF}{Vmyo \cdot 2} + \frac{(Ileak - Iup) Vnsr}{Vmyo} + \frac{Idiff Vss_sr}{Vmyo} \right)$$

$$\frac{d}{dt} Ca_ss_sr = (-bss_sr) \left(Idiff + Idiff_ss - \left(\frac{2 INaCa_ss_sr AF}{2 Vss_sr} + \frac{Irel Vjsr}{Vss_sr} \right) \right)$$

$$bss_sr = \frac{1}{1 + \frac{BSRmax \cdot KmBSR}{(KmBSR + Ca_ss_sr)^2} + \frac{BSLmax \cdot KmBSL}{(KmBSL + Ca_ss_sr)^2}}$$

$$\frac{d}{dt} Ca_ss_CaL = (-bss_cal) \left(\frac{ICaL AF}{2 Vss_CaL} - \frac{Idiff_ss Vss_sr}{Vss_CaL} \right)$$

$$bss_cal = \frac{1}{1 + \frac{BSRmax \cdot KmBSR}{(KmBSR + Ca_ss_CaL)^2} + \frac{BSLmax \cdot KmBSL}{(KmBSL + Ca_ss_CaL)^2}}$$

$$\frac{d}{dt} Ca_NSR = Iup - \left(Ileak + \frac{Itr Vjsr}{Vnsr} \right)$$

$$bcsqn = \frac{1}{1 + \frac{kmcsqn \cdot csqphar}{(Ca_JSR + kmcsqn)^2}}$$

$$\frac{d}{dt} Ca_JSR = bcsqn (Itr - Irel)$$

Figure 3.3. Screenshot of a model comparison result in the Cardiac Electrophysiology Web Lab. A user of the Web Lab can select a model to compare any two of its versions. The shown model (decker_2009.cellml) is available in two versions (*Decker 2009 - fixed* and *Decker 2009 - buggy*), as shown on the top of the page. Even though cardiac models are typically very complex (the figure only shows parts of the differential equations of one out of 43 components) it is not difficult to spot the differences reported by BiVeS, despite the change in the sequence of equations. The comparison is available online at s.binfal.de/decker-diff.

3.3. Versioning concept for a database

Traditionally, databases in the systems biology domain express relationships between entities in a simulation study as a means of joining tables. Thus, the number of allowed questions when searching for a simulation study is limited as the complexity of corresponding database queries increases rapidly. Moreover, neither models nor simulation descriptions or related data are easily representable as tabular structures. In contrast, NoSQL databases, and especially graph databases, are better suited to handle the largely heterogeneous data of simulation studies in terms of performance and flexibility. Masymos, a graph database for search and retrieval of computational models and related data, takes advantages of the NoSQL approach [HWW15]. However, this database does not respect versions of models, yet. In a bachelor thesis that I supervised we extended the database concept of Masymos to also store and track versions of models [Pet16].

3.3.1. Database structure implemented in Masymos

Masymos is a management platform for models and associated data. It is built on top of Neo4J, an ACID-compliant transactional database [Mil13; RWE13]. Masymos is able to integrate and store different kinds of data associated to a simulation study. It supports models encoded in either SBML or CellML, simulation descriptions in SED-ML format, and annotations using terms from different ontologies, such as SBO and KiSAO (compare Section 1.2.1). Masymos maintains several indices (e.g. over models, authors, annotations, etc.) to facilitate advanced searching and ranking [Sch13].

A schematic of the structure implemented in Masymos is shown in Figure 3.4. The figure indicates how a model of the cell division cycle [Tys91] encoded in both SBML (top left) and CellML (bottom left) is linked in Masymos. In Neo4J, nodes and relationships can be categorised using labels. For the SBML model, for example, Masymos creates a node labelled `DOCUMENT` representing the XML document (yellow) and a node labelled `MODEL` representing the model (light blue). The model node has multiple children: One reaction node labelled `SBML_REACTION` (red), three species nodes labelled `SBML_SPECIES` (green), and a compartment node labelled `SBML_COMPARTMENT` (brown). Furthermore, species may be linked to a reaction (e.g. `C2` and `CP` are linked to `Reaction3`), and species and reactions may be linked to a compartment (`pM` is linked to `Cell`). Almost every node in the graph can be linked to one or more annotations (grey). For example, the model node is annotated with the Kegg identifier `sce04111`¹³. Thus, it is semantically encoded that this model is about the cell cycle in yeast and, through that annotation, it will be connected to other models studying that same system. Please note, that Masymos' representation of that model is actually much more complex. In total, the model contains nine species and nine reactions. The model node alone is already linked to

¹³www.kegg.jp/entry/sce04111, accessed 5 July 2017

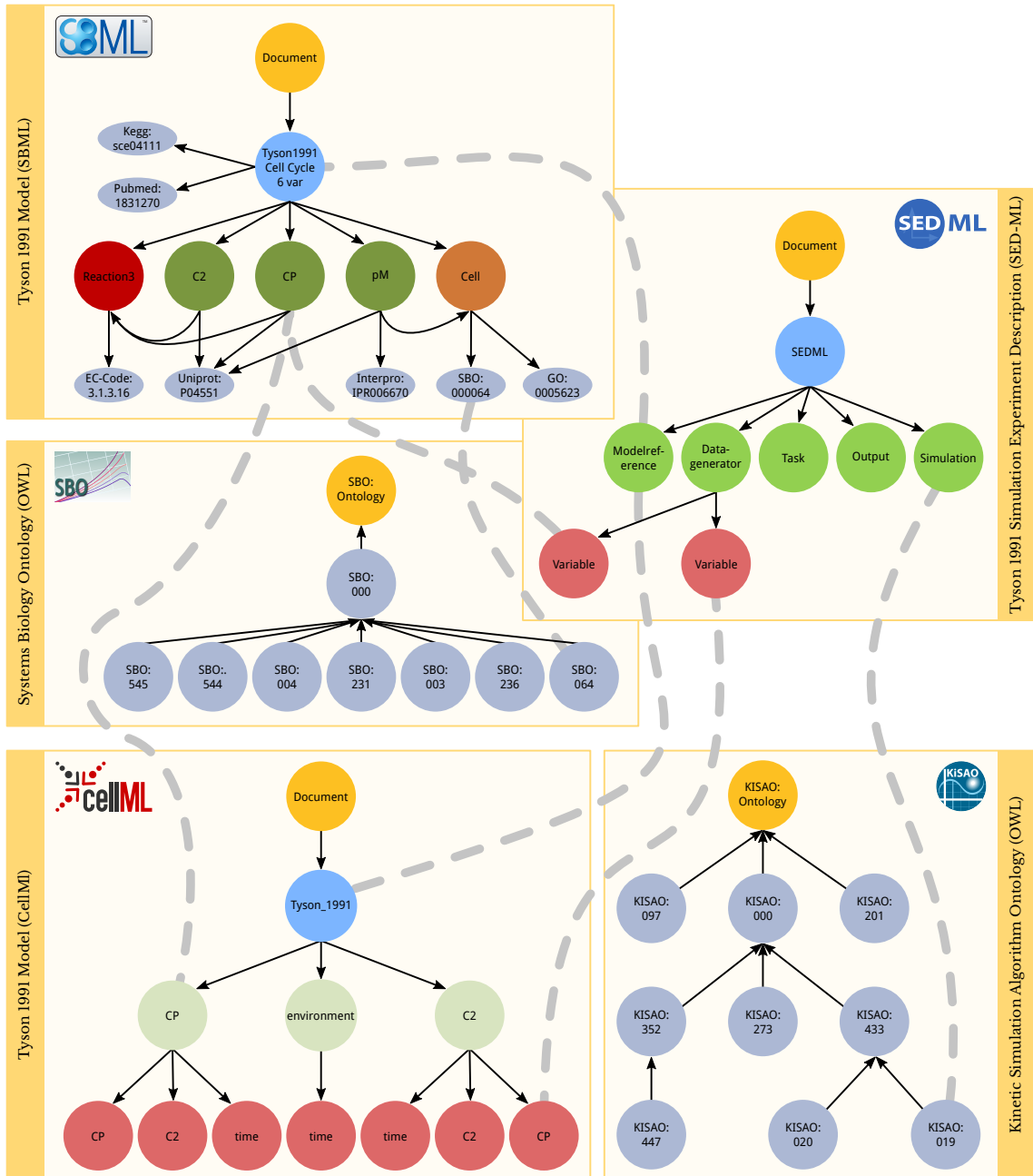


Figure 3.4. Database structure implemented in Masymos. The figure exemplarily shows five documents and how they are linked in Masymos: An SBML model (top left), a simulation description in SED-ML (top right), the SBO ontology (middle), a CellML Model (bottom left), and the KiSAO ontology (bottom right). Black arrows show relationships between entities of the same document. The grey dashed lines indicate how different documents are linked. For example, (i) the component CP in the CellML model and the Species in CP the SBML model represent the same biological entity, (ii) the Simulation entity of the SED-ML document is directly linked to the corresponding KiSAO term, and (iii) the SBO annotation of the compartment Cell in the SBML document is unambiguously resolved through the associated term SBO:064 in the ontology representation. The figure was rebuild from [HWW15] to improve the resolution and to fit it into this thesis.

Chapter 3. Applying the method for difference detection in systems biology projects

seven annotations, and the entities of this simple model are connected through hundreds of relationships. While top-down relationships in Masymos' model representation are typically very specific (e.g. model and reaction are connected through a relationship labelled `HAS_REACTION`), the nodes of the model hierarchy are also connected bottom-up through relationships with a label `BELONGS_TO`. Thus, given a random entity, it is easy to find the corresponding XML document by just following `BELONGS_TO` links.

However, Masymos defines no relationships to connect different versions of the same model. Moreover, there is currently no way to encode the changes between model versions. This emphasises the need for a method to properly handle multiple versions of a model.

3.3.2. Enhanced concept respecting model versions

Introducing a versioning concept for models to the Masymos database entails a number of challenges. First of all, versions of models need to be unambiguously identifiable. Second, the versions of a model need to be linked according to their evolution. Third, information about the changes between versions should be stored directly in Masymos, which would support users in searching for interesting versions.

Extensions of the database model and storage decisions

A new version of a model is saved in a new version of the document. Thus, versions were introduced on the level of documents. Additionally, two new relationships were implemented that can be used to link nodes of the type `DOCUMENT: HAS_PREDECESSOR` and `HAS_SUCCESSOR`. As the names suggest, the former relates from a newer version to an older version of the document and, vice versa, the latter connects from an older to a newer version. Moreover, a new property `VERSIONID` (carrying a unique value) was added to document nodes, so that versions of documents sharing the same `FILEID` can be distinguished. Consequently, models and their versions are unambiguously identifiable and their evolution can be represented through relationships.

Obviously, to also store differences the database structure needs to be extended more radically: A new label `DIFF` identifies nodes, which represent the collection of modifications between the linked documents. Exactly two nodes labelled `DOCUMENT` can be connected to a node labelled `DIFF` using `HAS_DIFF` relationships. To represent atomic differences a generic node label `DIFF_NODE` and four more concrete node labels `DIFF_INSERT`, `DIFF_DELETE`, `DIFF_MOVE`, and `DIFF_UPDATE` were introduced. A `DIFF` node may have an unlimited number of connections of label `HAS_DIFF_ENTRY` to nodes labelled `DIFF_NODE`. Nodes with a label `DIFF_NODE` should contain a list of properties in concordance with the machine-readable delta as exported from BiVeS, compare Section 2.2.5. Thus, these nodes will typically have properties such as `bives.newPath`, `bives.name`, or `bives.newValue` – of course depending on the characteristics of the specific change. Furthermore, a `DIFF_NODE` is connected to one or two nodes in a model entity. On the one hand, a relationship `IS_SOURCE` may link the difference to a model entity in the old version of the document (the predecessor).

Chapter 3. Applying the method for difference detection in systems biology projects

On the other hand, a relationship `IS_DESTINATION` may link the difference to a model entity in the new version of the document (the successor).

The described relationships of the database model are visualised in Figure 3.5. The model helps (i) to see which entities in a model have changed with respect to a previous/next version, (ii) to learn how many changes between two versions occurred, and (iii) to understand how model versions relate to each other. However, it remains difficult to filter for changes. For example, a user may want to see versions that changed the mathematical definition of a model. This user is then neither interested in corrections of typos nor in changes of the textual description of a model. Therefore, the COMODI ontology was integrated into Masymos. The terms of the ontology are connected through `isA` relationships in Masymos, compare Figure E.1 in the appendix. In addition, the special relationships `affects`, `appliesTo`, `hasReason`, and `hasIntention` were introduced, which can be used to link a `DIFF_NODE` to the a term of COMODI, characterising its impact. Furthermore, the relationship `DIFF_TRIGGERED_BY` may link two nodes labelled `DIFF_NODE`, according to the idea presented in Section 2.4.3. Using the COMODI ontology it is possible to attach meaning to the differences between versions of a model.

Implementation of the versioning concept in Masymos

Masymos' infrastructure consists of a database interface and a web-based API, both built on top of Neo4J. To realise support for model versions we extended the database interface and implemented aforementioned changes. For example, the methods to insert or delete a model (or model version) needs to generate and update `HAS_PREDECESSOR` and `HAS_SUCCESSION` relationships according to the models evolution. Moreover, a diff-plugin¹⁴ was developed to compare two model documents using BiVeS and to store the delta information in the graph database. This plugin integrates BiVeS as a library and uses its API to identify the differences. The delta produced by BiVeS is then stored in the graph according to the previously mentioned model. More specifically, for a comparison of a model document in versions V_1 and V_2 a node X of type `DIFF` and two relationships V_1 -[`HAS_DIFF`]-> X and V_2 -[`HAS_DIFF`]-> X are created. The new node X gets a few properties summarising the characteristics of the delta (such as the total number of insertions). Additionally, for every atomic change n between V_1 and V_2 a node Y_n labelled `DIFF_NODE` is created and connected to X via X -[`HAS_DIFF_ENTRY`]-> Y_n . According to the type of change Y_n gets a second label: `DIFF_INSERT`, `DIFF_DELETE`, `DIFF_MOVE`, or `DIFF_UPDATE`. Specific details on the change, such as old and new attribute values, are stored in Y_n as properties. Furthermore, the node Y_n is linked into nodes belonging to V_1 and V_2 representing the affected entity using the relationships `IS_SOURCE` (for entities in document V_1) and `IS_DESTINATION` (for entities in document V_2). As not every entity in a model has a direct representation in Masymos, the diff-plugin traverses the entity hierarchy of the model's XML tree in a bottom-up manner

¹⁴github.com/SemsProject/masymos-diff, accessed 7 July 2017

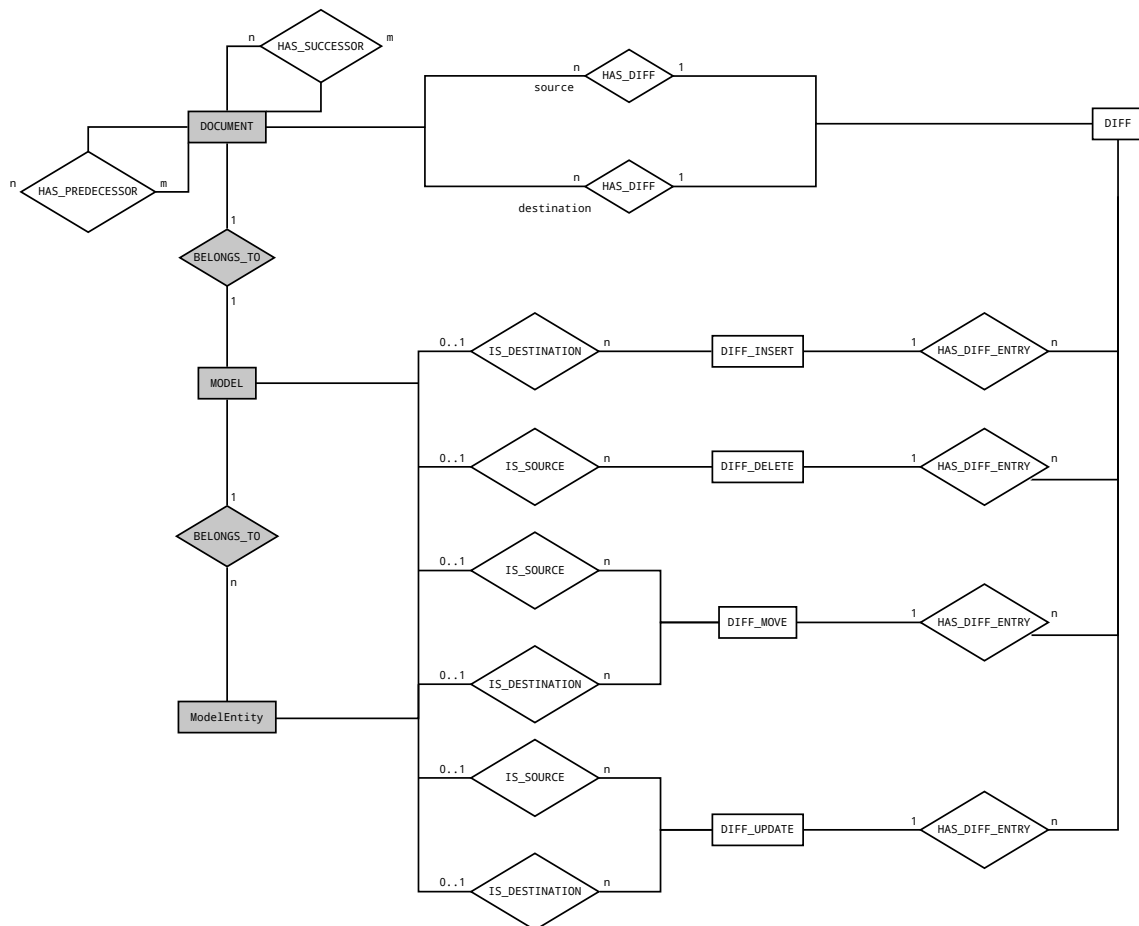


Figure 3.5. ER diagram of the extension on the database model implemented in Masymos. Nodes with a grey background are part of the original database model. Nodes with a white background are part of the extension. Documents may be linked using `HAS_PREDECESSOR` and `HAS_SUCCESOR` relationships, according to their evolution. Two documents may be linked to a node labelled `DIFF` using `HAS_DIFF` relationships. The `DIFF` node may contain any number of `HAS_DIFF_ENTRY` relationships to nodes labelled `DIFF_INSERT`, `DIFF_DELETE`, `DIFF_MOVE`, or `DIFF_UPDATE` (all being also labelled `DIFF_NODE`). Nodes labelled `DIFF_NODE` may furthermore be labelled to nodes belonging to a document using either `IS_SOURCE` or `IS_DESTINATION`. The figure was adapted from [Pet16].

Chapter 3. Applying the method for difference detection in systems biology projects

until it finds a parent entity, which is then linked to the change. A special property in a `DIFF_NODE` called `inherit` indicates if the corresponding change affects an entity directly (`inherit` is set to `false`) or if it affects the subtree of that entity (`inherit` is set to `true`). That is, changes in one term of the kinetic law of a reaction will be linked to the node representing the corresponding reaction in that model and its `DIFF_NODE`'s property `inherit` is set to `true`. Annotations with terms from COMODI as reported by BiVeS are directly translated into relationships between Y_n and the corresponding terms of the COMODI ontology in Masymos, compare Appendix E.2.

A second plugin for Masymos realises a RESTful interaction with the diff-plugin through HTTP connections. It can, for example, be used to trigger the comparison of models using the `/diff/service/trigger` endpoint. Once triggered, the diff-plugin will search for nodes V_1 and V_2 that (i) are labelled `DOCUMENT`, (ii) are related through V_1 –[`HAS_SUCCESSOR`]→ V_2 , (iii) both have a link to a node labelled `MODEL`, and (iv) do not have a relation to the same node labelled `DIFF`. For every tuple of V_1 and V_2 an asynchronous comparison task is scheduled which executes BiVeS and stores the differences according to the strategy explained above. These actions at the REST endpoint can, for example, be triggered (i) immediately when a new model version is added to the database, (ii) periodically, or (iii) manually, e.g., when installing the database.

3.3.3. Results and discussion

The extension of the database model is designed to assist modellers in discovering and understanding changes on models. In fact, using the new database model it is possible to implement a notification system that informs users in case of changes to their favourite models, e.g. models that a user shared or obtained through the database. Furthermore, this notification system may include filters to notify, e.g., modellers in case of changes in the mathematical definition and database maintainers in case of changes on the level of annotations.

The extended version of Masymos allows to encode the timeline of a model's development using `HAS_PREDECESSOR` and `HAS_SUCCESSOR` relationships. Models and versions are therefore unambiguously identifiable and accessible. Differences between versions can be encoded using novel labels for node and relationships. Knowledge about differences can be semantically annotated using terms from the COMODI ontology. The realisation of the database model, as visualised in Figure 3.5, is indicated in Figure 3.6. The figure shows Masymos' representation of two versions of a toy model and their differences. Nodes belonging to the delta are highlighted with a grey background. For example, the purple node with caption 1739 represents the `DIFF` node. Its properties describe the characteristics of the delta: `numUpdates`: 1, `numInserts`: 7, `numNodeChanges`: 2, `numTextChanges`: 0, `numMoves`: 0, `numDeletes`: 0. The yellow node with caption 1 is labelled with `DIFF_NODE` and `DIFF_NODE`. It represents the change of the initial concentration in species A from 100 to 120. As it affects the species directly, its `inherit` property is set to `false`. Its properties feature all details reported by BiVeS, as shown in Source Code 3.5. The green nodes with captions 2 to 6 represent the insertion of a species C to the model. While the node 2 inserts

```
1  "properties": {
2    "bives.id": 1,
3    "bives.newPath": "/sbml[1]/model[1]/listOfSpecies[1]/species[1]",
4    "bives.oldPath": "/sbml[1]/model[1]/listOfSpecies[1]/species[1]",
5    "bives.name": "initialConcentration",
6    "bives.oldValue": 100,
7    "bives.newValue": 120
8    "inherit": false,
9  }
```

Source Code 3.5. Properties of a `DIFF_NODE` in Masymos. The code shows the properties of the node with caption 1 from Figure 3.6, which corresponds to the update of the initial concentration of a species. The properties starting with `bives.` are reported by BiVeS.

the actual species node, nodes 3 to 6 equip the species with characteristic attributes: Node 3 inserts a link to the `compartment` which is identified by default, node 4 inserts its `id` with value `specC`, node 5 inserts its `initialConcentration` of 0, node 6 inserts its `name` with value `c`. Nodes 3 to 6 also get a relationship to 2 labelled `DIFF_TRIGGERED_BY` – together they can be understood as a group of changes that insert species `c` to the model version v_2 . Finally, the green nodes with captions 7 and 8 represent the insertion of species `c` as a product to reaction `R`. They both do not affect the reaction node directly, but its subtree (they create a new `speciesReference` in the reaction's `listOfProducts`). Therefore, the `inherit` property of both nodes is set to `true`. Both model versions of the model are artificially created for demonstration purposes. They are available from Appendix F.

Even though the newly introduced types of nodes and relationships representing differences between versions seem invasive, they are designed to not break previous algorithms and methods. They exclusively add information to a novel layer in the graph, which can easily be neglected. In contrast, the general idea of versions may influence existing search algorithms. If these algorithms are not developed with versions of models in mind, they may match several versions of the same model equally high for a given query. As a consequence, a search result may list the best matching model on the top 20 ranks – assuming the model has 20 versions – while the distinct versions may just differ slightly. This behaviour may justifiably be undesired. Existing algorithms should therefore be evaluated and updated accordingly.

This extension substantially increases the complexity of the graph. For example, in a graph of 2494 nodes with complete versioning information 55.97% of the nodes (1396) are related to storing versioning information (nodes labelled `DIFF` or `DIFF_NODE`). Indeed, that is a massive overhead. Size, however, does not play an important role as Masymos is just a central database, not meant for individual usage. Thus, we consciously trade storage space for search speed. However, depending on use case and preferences of database maintainers and users the space consumption can be decreased significantly with three adjustments. First, the `deltas` may only

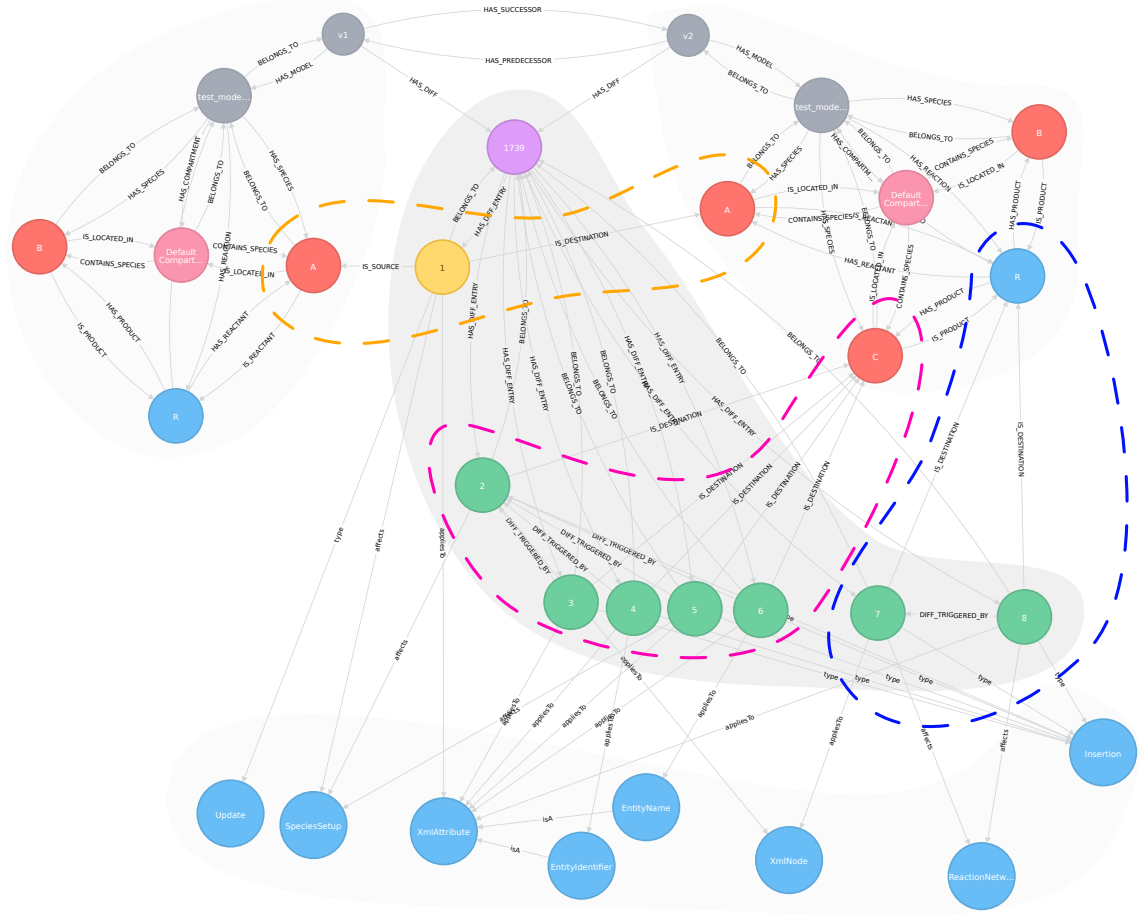


Figure 3.6. Representation of a delta in Masymos. The figure shows the graph of two versions of a model document (v_1 and v_2) in Masymos. The shown graph can be conceptually understood as four sub-graphs: (i) The top left shows the model version v_1 , (ii) the top right shows the model version v_2 , (iii) the middle shows the representation of the delta highlighted in a grey background, and (iv) the bottom shows associated terms of the COMODI ontology. The delta can again be divided into three different parts. (i) The dashed orange line shows the update of an attribute value in the species A. The yellow node with id 1 is labelled `DIFF_UPDATE` and connected to species A of v_1 using an `IS_SOURCE` relationship and to species A of v_2 using an `IS_DESTINATION` relationship. It is annotated with the terms `Update` and `SpeciesSetup` of the COMODI ontology. (ii) The dashed magenta line shows the insertion of a the new species C into version v_2 . The node with id 2 represents the insertion of the actual document node in the XML document. Nodes 3 to 6 are changes triggered by 2 – they represent the insertion of new attribute values, such as the species' name and its initial concentration. Therefore, the nodes 3 to 6 are each related to 2 using an edge labelled `DIFF_TRIGGERED_BY`. (iii) The dashed blue line shows the insertion of species C into the reaction R in version v_2 . Both, node 7 and node 8 affect the reaction network of the model and are annotated with the term `ReactionNetworkDefinition` of the COMODI ontology. The models in v_1 and v_2 were artificially created for this thesis. More information on both models is available from Appendix F.

Chapter 3. Applying the method for difference detection in systems biology projects

be computed on demand, i.e. when a user or tool needs the corresponding information. This, however, prevents large scale analysis and implies a serious increase in search durations if versions need to be compared beforehand. Second, details on the deltas may be reduced. For example, groups of changes can be condensed according to `DIFF_TRIGGERED_BY` relationships. The annotations of discarded nodes can be propagated accordingly to their condensed versions. In the example above, 750 of 1396 nodes are triggered by other `DIFF_NODES`. If removed only 646 nodes (out of 1396) are necessary to store the versioning information. Moreover, some of the details in the `DIFF_NODES`' properties are not necessarily useful for every use case. It may, for example, be justifiable to discard actual parameter values and XPath expressions, as the required information is linked through COMODI terms. For details on a delta it could still be regenerated on demand. Third, versions may be stored using a reverse-delta storage [BL97; Tic85], as common for traditional version control systems, such as SVN. That way, nodes common to multiple versions of the same model would only be stored once. This, in turn, means radical changes in the database model and, presumably, most of the existing algorithms, methods, and queries need to be adjusted.

In conclusion, the extension of the database model improves the transparency of model updates. It allows modellers to explore the evolution of models and permits improved search queries through advanced filters. Simulation description linked to a model may automatically be invalidated if, e.g., a necessary species was removed from a model. Furthermore, a notification system may alert users if a model they obtained is, for example, corrected or improved. I believe that this will ultimately foster model reuse.

3.4. Studying the evolution of open model repositories

A useful model is one that is being (re)used. The development of a successful model does not finish with its publication. During reuse, models are being modified, i.e. expanded, corrected, and refined. Even small changes in the encoding of a model can however significantly affect its interpretation. Thus, it is crucial to inform about changes that have occurred in a model. When reusing a well documented model, researchers save time, effort, and money [Wil+16]. However, a lack of transparent documentation of the conditions and boundaries applied to the model, as well as a lack of provenance information, can lower the trust in a model. In contrast, a transparent communication of changes in models increases their value [Sch+16]. To build an informative history about a model, all its versions need to be publicly accessible, and all changes across versions have to be well described [Wal+13].

Both BioModels Database and the Physiome Model Repository provide versions of published models through their websites. Access to raw version information allows to further process the data and to study model changes. My previously introduced method for difference detection, for example, helps researchers to compute and analyse the differences between two versions of

Chapter 3. Applying the method for difference detection in systems biology projects

a model [SW15]. Identified changes in model versions can then be classified using COMODI, an ontology of terms describing model changes [Sch+16].

Here I analyse raw model versions with respect to the frequency and influence of changes. Using the BiVeS tool, I identify the changes between all released versions of models available from BioModels Database and the Physiome Model Repository. I identify update patterns and provide an example of a model's history. The results show that models are indeed continuously subjected to changes. These changes, however, have different reasons, such as updates of the description format and error corrections. To demonstrate the impact of changes I explore the history of a Repressilator model from BioModels Database. My analysis demonstrates the numerous changes that occur in models. Even early models are still continually updated. I observed frequent improvements in annotations, which enriches the information one can gain from models. In order to provide interactive access to the changes in published models, I developed a freely available online platform.

3.4.1. Materials and methods

The data presented and analysed in this section was generated following the schematic shown in Figure 3.7. The heart of my pipeline is the Java tool *Statistics Generator*¹⁵ (SG), which wraps the *ModelCrawler*¹⁶ and BiVeS to obtain and process the data. It first runs the ModelCrawler to retrieve all available model versions from *BioModels Database* and the *Physiome Model Repository*. The SG then uses BiVeS to calculate the differences between every subsequent version of each model. Afterwards, BiVeS' output is evaluated and the results are stored in separate *data tables*. Based on these tables, a set of R scripts generates static figures, and the *ModelStats* (MoSt) website provides interactive visualisations of the data. The SG is available as a Docker image¹⁷. It can be used to regenerate the data tables. A set of R¹⁸ scripts are available through the source code of the SG¹⁹. The MoSt website²⁰ can be installed and run by everyone; the source code is available from GitHub²¹.

The Data

The data used to generate the figures originate from BioModels Database and the Physiome Model Repository. BioModels Database provides curated and non-curated models in SBML format. New models are submitted to the non-curated branch. Once the modelling results could be reproduced by a curator, the model moves to the curated branch. Release 31 of BioModels Database contains 640 models in the curated branch and 1000 models in the non-curated

¹⁵github.com/binfalse/BiVeS-StatsGenerator, accessed 24 October 2017

¹⁶github.com/SemsProject/ModelCrawler, accessed 24 October 2017

¹⁷hub.docker.com/r/binfalse/bives-statsgenerator, accessed 24 October 2017

¹⁸www.r-project.org, accessed 24 October 2017

¹⁹github.com/binfalse/BiVeS-StatsGenerator/tree/master/src/main/resources, accessed 24 October 2017

²⁰most.bio.informatik.uni-rostock.de, accessed 24 October 2017

²¹github.com/SemsProject/MoSt, accessed 24 October 2017

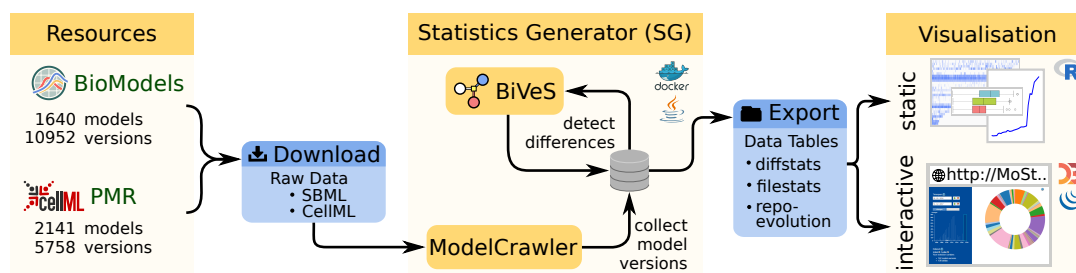


Figure 3.7. Pipeline used to obtain, analyse, and visualise the data. First, all relevant models and their versions are downloaded from BioModels Database and from the Physiome Model Repository (PMR) using the ModelCrawler. BiVeS detects and reports the differences between consecutive model versions. The Statistics Generator (SG) then exports the results as data tables, which collect statistics on models and changes. The data tables are used in R scripts and in the MoSt website to produce static and interactive visualisations, as presented in this paper.

branch. For this study we considered all models in all release versions since the launch of the repository in April 2005 and the time of writing in July 2017 (10 952 model files).

The Physiome Model Repository provides curated and non-curated models, primarily in CellML format. The models are embedded in workspaces, which may contain further model related data, such as network visualisations, simulation descriptions, and links to previous versions of the studies. Particularly interesting and well documented revisions of workspaces can be published as exposures [Mil+11]. Workspaces may contain multiple models, which may be decomposed into different documents. For our study we treated every valid CellML document as a CellML model. For this work all 2782 models files from 651 publicly available workspaces were retrieved.

ModelCrawler: Acquiring models and versions

The ModelCrawler is a Java tool that retrieves models from open model repositories. It currently implements two modules: One for BioModels Database and one for the Physiome Model Repository. When retrieving data from BioModels Database, the ModelCrawler mirrors the corresponding FTP server at the EBI²², extracts the models of each release, and stores them locally. When retrieving data from the Physiome Model Repository, the ModelCrawler iterates through the list of public workspaces²³, clones the corresponding Git repositories, extracts the models in all revisions, and stores them locally. In addition, the ModelCrawler collects and stores metadata, including information about the model's origin and time stamps for each model version. For BioModels Database, the time stamps correspond to the release date of the database. The Physiome Model Repository provides precise version information through their repository back-end (git-log) [Mil+11].

²²<ftp://ftp.ebi.ac.uk/pub/databases/biomodels/releases/>, accessed 24 October 2017

²³models.physiomeproject.org/workspace_list_txt, accessed 24 October 2017

BiVeS: Comparing versions of a model

BiVeS compares the retrieved model versions and identifies the differences in the XML representation [SW15]. The tool distinguishes four types of changes (*insertion*, *deletion*, *move*, *update*) and three different kinds of entities in an XML document (*element node*, *attribute node*, *text node*) that are subjected to changes. BiVeS computes the differences between every two consecutive versions of each model retrieved by the ModelCrawler. In total, BiVeS generated 12 467 deltas between model versions.

Statistics Generator (SG): Evaluating the BiVeS output

The results of BiVeS' computation are post-processed and aggregated into three data tables.

The first table contains details about the models files in all available versions (*filestats*). Each row stores the number of (i) XML nodes, (ii) species, (iii) reactions, (iv) compartments, (v) functions, (vi) parameters, (vii) rules, (viii) events, (ix) units, (x) variables and (xi) components in the model. Additionally, information about the curation status of the model, encoding format, identifiers for model and version, and the URL to the model file is collected.

The second table contains data about the evolution of the repositories (*repo-evolution*). Starting from April 11th 2005 (BioModels Database emerged), it stores the number of models in BioModels Database and in the Physiome Model Repository. For each point in time, the details of the models, see (i)-(xi) above, are accumulated into three feature vectors. One vector for BioModels Database, one for Physiome Model Repository, and another one representing both repositories combined.

The third table contains details on the differences between two successive versions of a model (*diffstats*). Every version transition is examined with both the Unix *diff* tool (inserts and deletes) and BiVeS (inserts, deletes, updates, moves, and triggered operations [SW15]). Furthermore, each row in the table contains the corresponding model identifier and the identifiers for both versions of the model. Thus, every entry in the *diffstats* table can be linked to the model versions in the *filestats* table.

Generating the static figures

Figures 3.8, 3.9, 3.10 were generated using aforementioned R scripts²⁴. Figure 3.8 shows how the repositories evolve over time (number of models, size of the models). Figure 3.9 shows how frequently models are updated and how significant the changes are. Figure 3.10 shows the different types of changes and affected parts in the model document. Figure 3.11 was generated by an extra module implemented in the SG. It is based on an SVG template derived from the COMODI ontology [Sch+16] and visualises types and targets of changes.

²⁴github.com/binfalse/BiVeS-StatsGenerator/tree/master/src/main/resources, accessed 24 October 2017

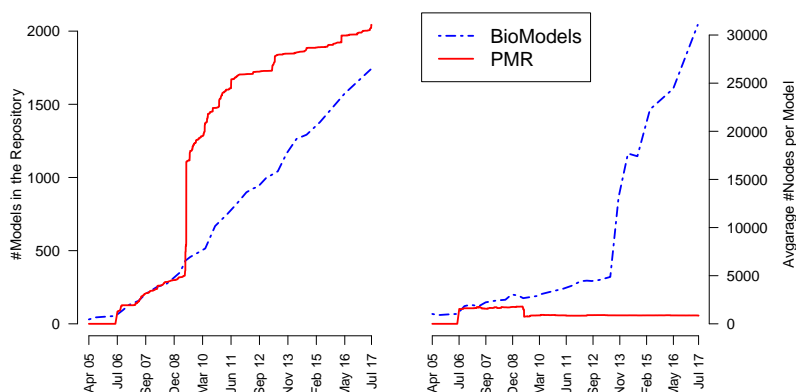


Figure 3.8. Number and size of models in BioModels Database and in the Physiome Model Repository. The plot shows the total number of models (left) and the mean number of nodes per model (right) in BioModels Database (dotted line) and in the Physiome Model Repository (solid line) since the launch of the databases until July 2017.

MoSt: Interactive visualisations

The ModelStats website (MoSt) at most.bio.informatik.uni-rostock.de allows for interactive visualisations of the data presented here. The portal makes active use of JavaScript libraries such as JQuery²⁵, D3 [BOH11], and highlight.js²⁶ to provide intuitive access to the model evolution in open repositories. It integrates the DiViL²⁷ tool to visualise the differences between reaction networks.

3.4.2. Results

In this section I analyse the evolution of SBML and CellML models from BioModels Database and the Physiome Model Repository. Overall, I studied 3781 models, with a total of 16 710 model versions. The results were obtained after applying the pipeline described in Figure 3.7.

Trends in model repositories

Figure 3.8 shows number and size of reusable models in the Physiome Model Repository and BioModels Database. The left panel verifies a steady increase in the number of models in both repositories. The right panel furthermore reveals a significant increase in average number of nodes per model for BioModels Database. This observation confirms previous observations in the literature [CLN13; Hen+10]. The first heavy increase appears in June 2013, when the average number of nodes rises from 4866 to 13118 nodes per model. This increase is due to the publication of a large SBML model encoding the global reconstruction of human metabolism (Recon2, [Thi+13]). A second increase can be observed in February 2014, when

²⁵jquery.com, accessed 24 October 2017

²⁶highlightjs.org, accessed 24 October 2017

²⁷github.com/Gebbi8/DiViL, accessed 24 October 2017

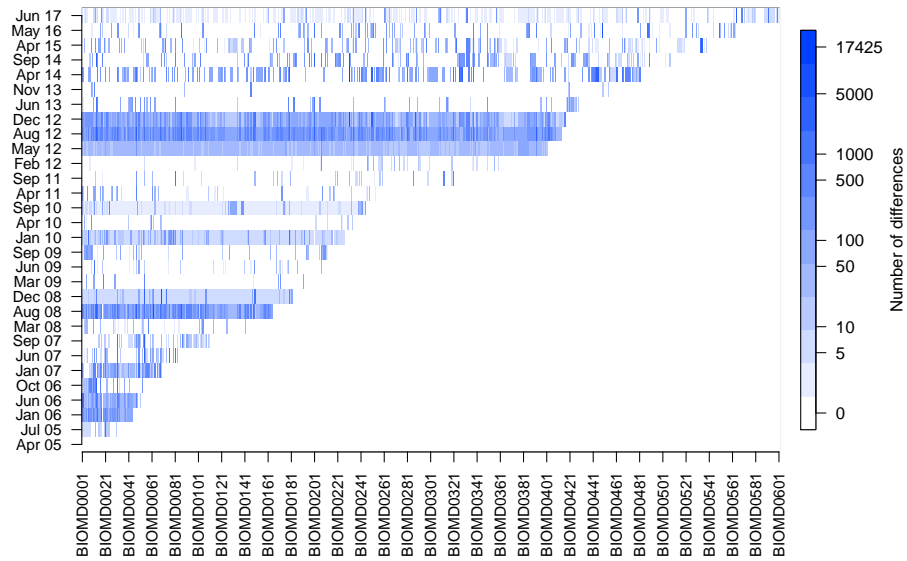


Figure 3.9. Updates of models in BioModels Database. The plot shows every recorded update of each model in the curated branch of BioModels Database. The rows (y-axis) show changes per official release of BioModels Database. The columns (x-axis) represent model files. Whenever a model was updated, a small blue vertical line indicates how many changes BiVeS detected between the old and new version of the model. Dark blue indicates many changes (maximum of 17 425 operations), light blue indicates few changes (minimum of 0-5 change operations).

the SBML encoding of a genome scale metabolic model was published [Mar+13]. Surprisingly, the average number of nodes remains stable for the Physiome Model Repository. As of July 2017, the average number of nodes per model is 31 059.1 for BioModels Database and 863.6 for the Physiome Model Repository.

Frequency of updates

Figure 3.9 visualises model updates in BioModels Database. Each coloured point indicates a change of a model in a specific release. The colour intensity reflects the number of changes: The darker the colour, the larger the number of modifications. Besides proving that models are subject to changes, the figure also reveals interesting patterns: horizontal blue bars indicate that some releases affect the majority of models. For example, the updates in December 2008 can be explained by the newly included instructions in every model on how to cite BioModels Database. The blue line in May 2012 can be explained by a change in BioModels Database’s legal terms: all models were published under the terms of the *CCO Public Domain Dedication*; their *notes* section was updated accordingly.

Another set of updates relates to the SBML annotation scheme. In June 2006, the introduction of qualified references to external resources [Nov+05] affected all annotated models. In August 2012, URNs in the annotations were replaced by links to *identifiers.org*, causing another major update of the database. In June 2017, the BioModels Database team revised the annotations in a majority of their models. For example, they annotated many models with terms from GO,

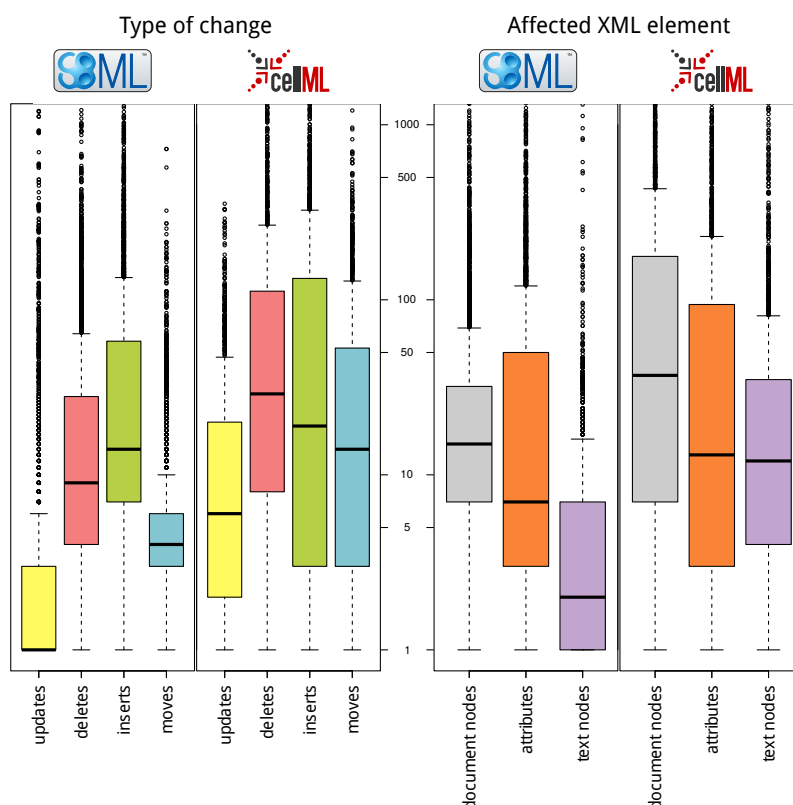


Figure 3.10. Types of diff operations. The boxplots quantify the types of changes (left boxplots) and affected components in the XML documents (right boxplots) in BioModels Database (boxplots 1 and 3) and the Physiome Model Repository (boxplots 2 and 4).

renamed qualifiers (e.g. `bqbiol:occursIn` to `bqbiol:hasTaxon`), and updated the timestamps of modifications.

As of July 2017, I observe an average of 4.49 versions per model in its first five years after publication. BiVeS reports an average of 327.92 differences between two subsequent versions of a model. However, not all changes do necessarily influence the behaviour of the model. Some are due to format updates, to design changes, or to changes in the model annotation [Sch+16].

Delta composition and characterisation of changes

Figure 3.10 quantifies the delta compositions. The left-hand side shows the type of changes in SBML (boxplot 1) and CellML (boxplot 2) documents, respectively. I distinguish four types of changes: *inserts*, *deletes*, *updates*, and *moves*. The majority of changes are inserts and deletes; there are just a few updates. Tendentially, there are more differences between versions of a CellML document. More specifically, I can see that entities in the CellML documents move more frequently than the ones in SBML documents.

The models considered in the study are all encoded in the Extensible Markup Language (XML). XML supports the concepts of elements (*document nodes*), attributes that further describe elements (*attributes*), and human-readable pieces of text (*text nodes*). Both SBML and CellML

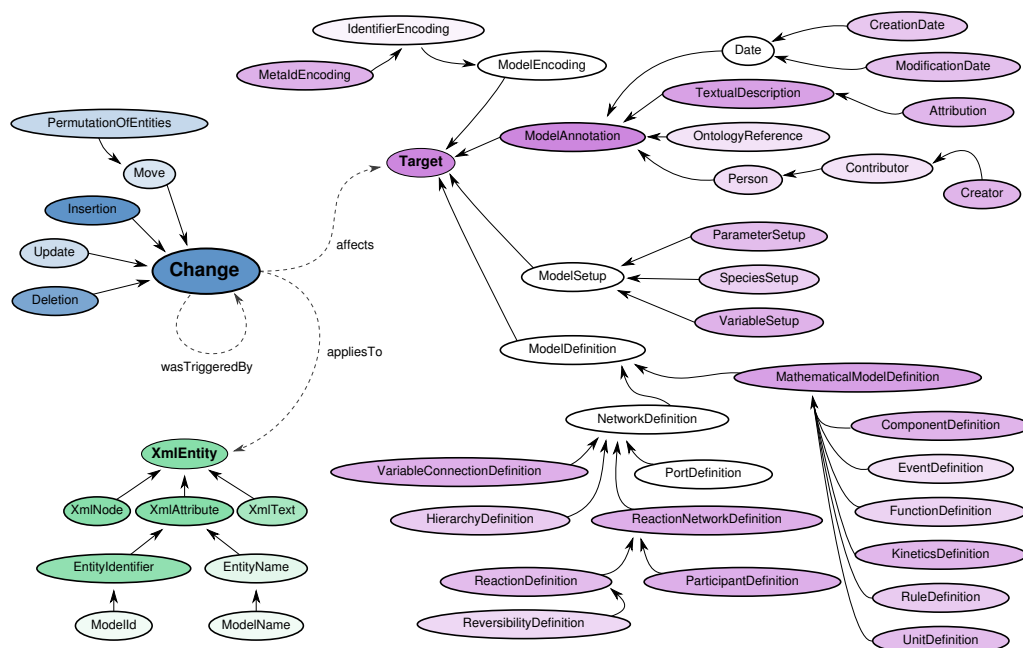


Figure 3.11. Coverage of COMODI terms. Colours are scaled individually. Thus, it is not possible to derive a quantitative statement between nodes of different colours. Only terms from the same branch can be compared.

are derivatives of XML and define the basic structure of a model using *document nodes*. Attributes store further information about model entities. In SBML, for example, a biological entity is represented by a *document node* which may then contain *attributes*, such as an `initialConcentration`. Both formats use *text nodes* to, for example, store meta information about a model or its entities. The right-hand side of Figure 3.10 shows that the least modifications in SBML documents affect the *text nodes*. Most updates in CellML documents affect *document nodes*, while the frequency of changes on *text nodes* and *attributes* are relatively similar.

The decision whether a change is relevant or not cannot always be made automatically. However, it helps to determine where a change takes effect in the model. Using the COMODI ontology, information about the characteristics can be described semantically. The pipeline is able to annotate changes with a subset of the COMODI ontology. However, it is not able to derive information about the intention nor the reason of a change. Figure 3.11 shows the branches of the COMODI ontology coloured in blue (Change), purple (Target), and green (XmlEntity). The intensity of the colour indicates how often a difference has been automatically classified with the associated term. For example, it is apparent that the terms *Insertion* and *Deletion* are darker than *Update* and *Move*, which is in concordance with Figure 3.10.

The ModelStats website

To provide an interactive access to the evolution of computational models we developed the MoSt web tool (most.bio.informatik.uni-rostock.de). MoSt provides a number of filters. It is,

Chapter 3. Applying the method for difference detection in systems biology projects

for example, possible to specify a time range or the model format. Furthermore, the data can be filtered for specific model identifiers. Thus, the evolution of a single model or a subset of models can be analysed. Repositories may link from a model's page to information about its evolution in MoSt. For example, most.bio.informatik.uni-rostock.de/#m:BIOMD0000000012,v:d,d1:2010-06-01,d2:2011-04-30 shows the evolution of model BIOMD0000000012 between June 1st 2006 and April 30th 2011. Furthermore, most.bio.informatik.uni-rostock.de/#m:BIOMD,v:d filters for all models whose identifier start with BIOMD, effectively selecting all curated models from BioModels Database²⁸

MoSt features four types of visualisations. A donut chart visualises each model transition in the selected time range; the amount of changes in a version transition is reflected by the size of the donut's slice. A heatmap provides more details about the actual numbers of *diff* operations for each transition; the height of a heat bar corresponds to the total number of changes. Both visualisations are interactive and provide access to more details on the changes between two versions of a model. The differences can be recomputed online using the BiVeS web application and the results are shown (human-readable report, graphic, XML encoded differences, COMODI terms). Finally, MoSt offers two boxplot visualisations, similar to Figure 3.10. One boxplot visualises the type of change (*move*, *insert*, *delete*, or *update*) of each model transition within the selected time range. The other boxplot shows which parts in the XML documents were subject to change (*text nodes*, *attributes*, or *document nodes*). Taken together, the MoSt tool allows researchers to explore the history of models in BioModels Database and the Physiome Model Repository.

3.4.3. Example: Changes in the repressilator model

The classical example of the Repressilator [EL00] showcases how a model in BioModels Database changes over time and how my tools contribute to a better understanding of these changes. The Repressilator is a synthetic model that links three transcriptional repressors to build an oscillating network. Its practical applicability has, for example, been shown for multiple organisms, including *Arabidopsis* [Pok+12] and *Escherichia coli* [EL00]. The model was first released in BioModels Database in September 2005²⁹ and is identified by BIOMD0000000012. In total, the SBML document has been modified 21 times. The model homepage at BioModels Database³⁰ already provides a textual description for many of the changes.

Figure 3.12 displays six versions of the Repressilator model in SBGN (Systems Biology Graphical Notation), a standard to visualise biological networks as graphs [Nov+09]. The figure also highlights the differences between the versions as identified by BiVeS: (i) elements

²⁸More information on filters and the different visualisations is available from MoSt's project page at github.com/SemsProject/MoSt, accessed 19 October 2017.

²⁹biomodels.bio.informatik.uni-rostock.de/BIOMD0000000012.xml/2005-04-11/BIOMD0000000012.xml, accessed 3 October 2017

³⁰BIOMD0000000012, <http://identifiers.org/biomodels.db/BIOMD0000000012>, accessed 3 October 2017

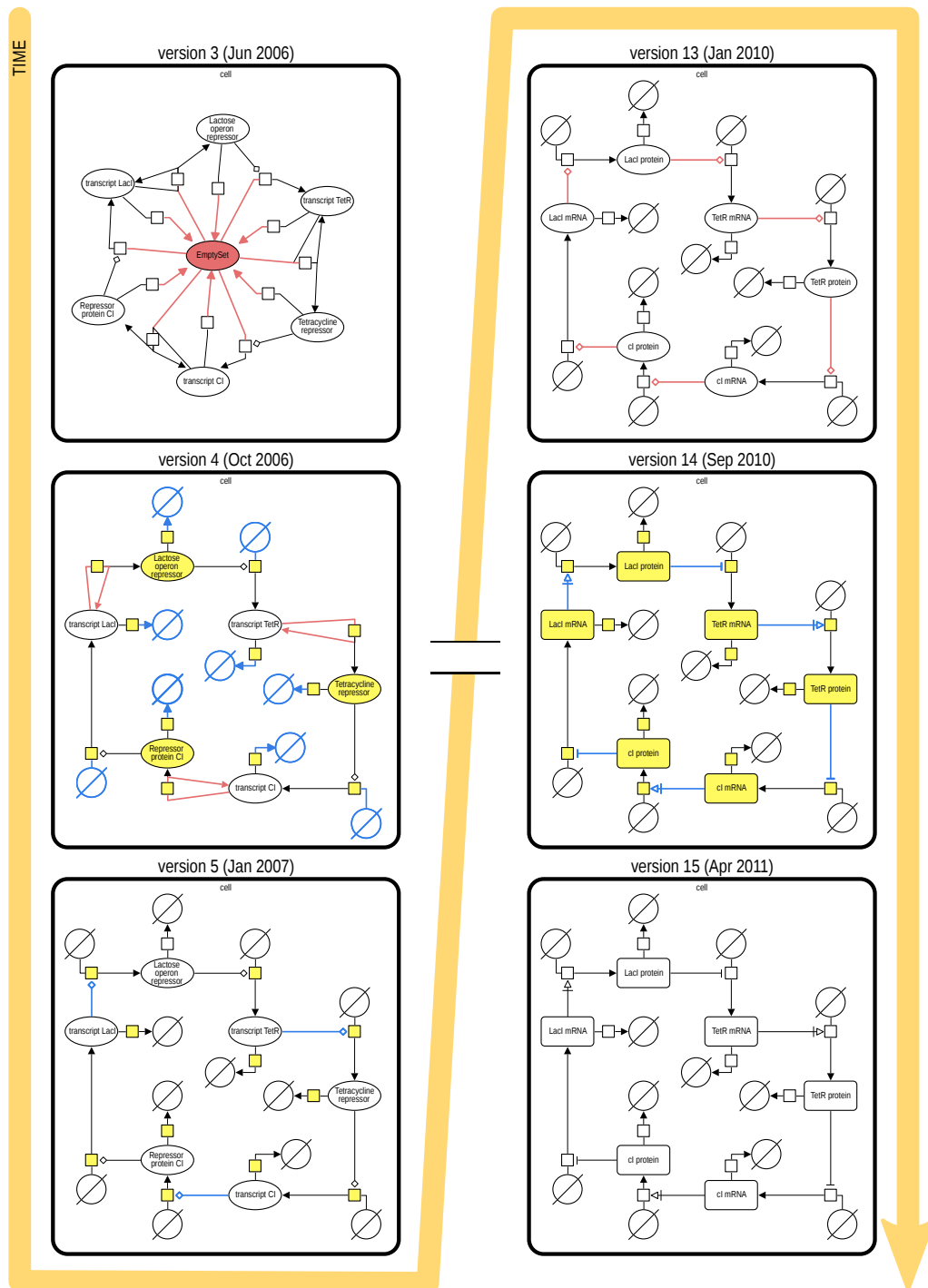


Figure 3.12. Differences between versions of the Repressilator model (BIOMD0000000012) in BioModels Database. Six versions of the Repressilator model (versions 3, 4, 5, 13, 14 and 15) are visualised in SBGN (generated using VANTED, [Roh+12] and SBGN-ED [CKS10]). The differences between the versions, as identified by BiVeS, are highlighted with a colour code: Updates are in yellow, inserts are in blue and deletes are in red.

Chapter 3. Applying the method for difference detection in systems biology projects

that have been removed in the subsequent version are coloured in red, (ii) elements that have been introduced in the current version are coloured in blue, and (iii) changes, which are not visible in the reaction network (e. g. updates of an initial concentration) are coloured in yellow. Please note that versions 3 to 5 (left column) and versions 13 to 15 (right column) are consecutive, while there is a time leap between versions 5 and 13 (left and right column).

The first transition displayed in the figure shows the deletion of one SBML entity (`emptySet`) between version 3 and version 4. This change is caused by a design decision on the SBML level. It does not affect the biological meaning, but it changes the SBML file and has a significant effect on the SBGN representation.

The second transition between version 4 and version 5 comprises of updates (in yellow) and changes to the reaction network (in blue and red). Specifically, the modifications rectify the effect of the transcripts over the translation of the repressors. Version 5 of the model encodes for the fact that the repressor is not created from the transcript (version 4), but that the transcripts modulate the translation of the repressors (version 5).

The third transition between version 13 and version 14 mainly improves the annotation of the model entities, reflected by a more detailed SBGN map. For example, the encoding of arrows and glyphs is more specific in version 14 (e. g., species are marked as macromolecules, *TetR protein* is described as an inhibitor for the translation of *ci mRNA*). Please note that the names of the species were updated at some point between version 5 and version 13 (not highlighted in the figure).

Finally, the fourth transition between version 14 and version 15 does not show any changes in the model. The reason for the existence of version 15 is that BioModels Database generates a new model version for every model at each release.

3.4.4. Discussion

Models are continuously subjected to changes. To understand the impact, characteristics, and frequency of changes I analysed the evolution of simulation models in open model repositories. The data presented in this paper has been generated following the pipeline described in Figure 3.7.

Many changes do not affect the model in the biological sense. When studying the similarity or changes in two versions of a model, different aspects may be considered [Wal+16]. Depending on the actual use of the model, some aspects are more relevant than others. In the Repressilator model, for example, the first transition (between versions 3 and 4 in Figure 3.12) affects the SBML encoding of the model, but not the biological system. Similar examples are updates in the SBML specification, updated publications, or new reference schemes to external data sources. These changes, if applied to a repository, affect the majority of models (see again blue bars in Figure 3.9). They typically do not affect obtained results. The knowledge about these changes can still be relevant for developers implementing tool support for SBML and CellML.

Chapter 3. Applying the method for difference detection in systems biology projects

However, researchers looking for changes in the biological model definition may exfiltrate the changes behind the horizontal blue bars in Figure 3.9. Annotations with terms from the COMODI ontology support users in distinguishing between relevant and unimportant changes.

I also observed that some models are changed more often than others. I was not able to determine whether “famous” models are updated more frequently (e. g., because they are checked by more scientists) or less frequently (e. g., because one reason for their frequent reuse is their quality). This analysis, however, could be an interesting endeavour for an experienced modeller. With respect to encoding formats, the data suggests that changes in CellML models are more radical in comparison to changes in SBML models, see Figure 3.10. However, significant changes can be expected with the implementation of SBML Level 3 models, which have a fundamentally different structure, and may import different SBML constructs from the so-called packages [Huc+10].

The updates with release 31 of BioModels Database in June 2017 do not seem very invasive when looking at Figure 3.9. However, when comparing the figure with MoSt’s filter graphic, one might speculate a discrepancy and hypothesise the large amount of changed files come from the Physiome Model Repository. This is not the case, though. The majority of the 1250 updated documents origin from BioModels Database. More specifically, 367 models from the curated branch and 640 models from the non-curated branch were affected. That means, minimal changes were introduced in about half the models from the curated branch. And indeed, the few resulting light-blue bars seem unsuspicious in Figure 3.9. They are, however, very prominent in MoSt’s time-slider, which displays the number of new versions introduced at a point in time. This suggests a curation initiative at BioModels Database, which affected a significant amount of their models.

The figures presented in this section are based on the state of BioModels Database and the Physiome Model Repository as of July 2017. They provide a global view on changes in models in the past years. However, during my investigation I observed that the history of model changes is not stable. It can happen that the history of releases in the repository is changed. I found two possible explanations for this. First, the Physiome Model Repository works on the basis of so-called workspaces. When a new workspace becomes public, the whole history of that workspace is also published, thereby slightly rewriting the (publicly visible) history of the whole repository. Second, I observed that the latest releases of BioModels Database can be updated up to the point when a new release is published, leading to inconsistencies in the latest data sets. Hence, it is important to remember that the figures may change in the future and yet affect shown data from the “past”.

My investigations do not provide information about who introduced a change in a model. This information is difficult, and sometimes impossible, to retrieve. For example, in BioModels Database one cannot see who changed a file; only snapshots of the repository are openly available. I want to encourage the maintainers of repositories to provide a system where curators

Chapter 3. Applying the method for difference detection in systems biology projects

and modellers can transparently track the evolution of a project, e.g. using PROV-O [MLS13] to encode the provenance and COMODI to describe reason, intention, and effects of a change. If, in future, model repositories implement such a system, incorporating that data will be an interesting extension to MoSt.

While this section provides only a global view on available models at a certain point in time, the web portal MoSt can be used to filter the models by id, time, or format. Thus, a personalised exploration of model histories is possible. MoSt is a static web project and open source available at GitHub³¹. Thus, it is easy to create a new instance of MoSt, which allows for more control and flexibility. In addition, the generated data tables are accessible through MoSt's web page. MoSt, in turn, is regularly being updated to reflect the latest state of BioModels Database and the Physiome Model Repository. Everyone is encouraged to extend MoSt with further useful analyses.

3.4.5. Conclusions

The reuse of models is still impeded by a lack of trust and documentation. A detailed and transparent documentation of all aspects of the model, including its provenance, will improve this situation. Knowledge about a model's provenance can avoid the repetition of mistakes already faced by others. More insights are gained into how the system evolves from initial findings to a profound understanding. It is the responsibility of the maintainers of model repositories to offer transparent model provenance to their users.

In this work, I evaluated publicly available versions of models in BioModels Database and the Physiome Model Repository, and I searched for irregularities and interesting pattern in the plots. The results inform scientists on how models evolve. As some changes affect the biological network, one conclusion drawn from this work is that existing models should continuously be monitored for changes. The web tool MoSt provides interactive access to model changes and displays the actual differences between single model versions.

³¹github.com/SemsProject/MoSt, accessed 1 October 2017

CHAPTER 4

SUPPORT FOR SHAREABLE AND REPRODUCIBLE SIMULATION STUDIES

Sharing simulation studies is essential for the advance of research in computational biology. Simulation studies typically consist of multiple files in fragile relations. For simulation studies to be reproducible, researchers need to share all the data and relations. Therefore we invented the COMBINE archive which eases the management of files related to a simulation study, fosters collaboration, and ultimately enables the exchange of reproducible simulation studies.

This chapter is based on five publications that support sharing reproducible simulation studies. The idea of COMBINE archives was published as “*COMBINE archive and OMEX format: One file to share all information to reproduce a modeling project*” in BMC Bioinformatics [Ber+14]. I published a use case that demonstrates the impact of COMBINE archives on dynamical modeling of syncytial mitotic cycles in *Drosophila* embryos in F1000Research [SW16a]. I have been involved in the development of tools published as original papers [Ber+17] or conference contributions [Sch+14a; Sch+14b], which support the usage of COMBINE archives. A novel method published in “*Extracting reproducible simulation studies from model repositories using the CombineArchive Toolkit*” [SW15] shows how tools can be tied together to publish reproducible simulation studies. Finally, decomposed simulation studies can be benchmarked and compared using “*The Cardiac Electrophysiology Web Lab*”, which I published in the Biophysical Journal [CSM16].

4.1. Challenges with the reproduction of modelling results

Computational modelling has become an indispensable tool in systems biology and systems medicine [Fin+04; GH13]. The steadily increasing size and complexity of simulation studies pose additional challenges to sharing reproducible results [WW16]. The ability to obtain similar results when reproducing an experiment is a tenet of modern science. The reproducibility of a scientific study depends on the careful description of the original experiment, including the methods and tools used to perform the experiment, the substrate on which to perform the experiment, and the precise experimental setup, including all necessary influences from the environment. When the result is meant to be presented after post-processing, it is also imperative to provide the details of the processing steps. Repeated mentions of problems with replication and reproducibility [BE12; Ioa+09; PSA11] led to the development of new standards, tools, and methods for the transfer of reproducible simulation studies [Bec+10; Ber+14; Cor+12; San+13; Sch+14b]. The *Computational Modelling in Biology Network* (COMBINE) coordinates the development of standard formats for various aspects of a simulation study: The Systems Biology Markup Language (SBML) [Huc+03] and CellML [Cue+03] encode the mathematical models; the Systems Biology Graphical Notation (SBGN) [Nov+09] encodes the visual representation of models; the Simulation Experiment Description Markup Language (SED-ML) [Wal+11b] encodes the simulation recipes; the Systems Biology Result Markup Language (SBRML) [Dad+10] and the Numerical Markup Language (NuML) encode numerical data and simulation results. Further more, other fields in the life sciences, such as ecology [Gri+06] and drug development [Swa+15], have also started to develop standards to encode their specialised mathematical models. This, however, emphasises how today's studies consist of multiple, heterogeneous, and sometimes distributed data files, leading to the challenge of exchanging complete and thus reproducible results. Even though, standards exist to encode modelling-related data, which may be publicly available in open repositories, all the files necessary for the description of a model must be exchanged in order to fully understand and use the model.

Managing and sharing multiple files entails a number of difficulties and can be tedious and error-prone. Some files necessary to build or process a model might have moved, or even be deleted, prohibiting the reproduction of results and ultimately even reusing the model. In addition, models are not static but evolve over time. Some changes, such as corrections of parameter values or modifications in the network structure, affect the outcome of a scientific study. Corrections of previously published model code must be communicated and propagated to all instances of the model that have been reused in other studies. It is therefore necessary to specify the version of a model that has been used in a simulation [Wal+13].

Several projects and initiatives, such as COMBINE (co.mbine.org), FAIRDOM (fair-dom.org), and the Reproducibility Initiative (reproducibilityinitiative.org), already dedicate their efforts to

reproducibility issues. Solutions towards improved management of computational models have recently been proposed, involving better accessibility of models and versions thereof, and better links between all files related to a specific study [Li+10; Mil+11; Wal+13; Wol+11]. However, these approaches focus on long-term availability and management of modelling experiments. Instead, modellers need means for easy export and exchange of simulation studies, as provided by an archive. A major advantage of an archive is that a single file can encapsulate everything there is to know about a specific modelling project, including the instructions on how to “open” the archive and interpret it. Similar examples from the domain of computer science are the packages of Debian GNU/Linux¹ and Red Hat² based operating systems, the Java Archives³, the electronic book format⁴, the Microsoft Office Open XML⁵, and the Open Document Formats⁶. Two archive formats that are applicable in the life sciences are Research Objects [Bec+10] and COMBINE archives. While the former is very generic, the latter is specifically designed for simulation studies encoded using COMBINE standards. Therefore, I will focus on COMBINE archives.

4.2. COMBINE archive and OMEX format

The SED-ML community initially developed the concept behind the COMBINE archive, when faced with the need to encapsulate the simulation experiment description and the models used to perform the corresponding *in silico* experiment [Wal+11c]. Members of the community extended the aim of the SED-ML archive to encompass any file type that would be useful during a modelling and simulation procedure, and wrote an initial technical description. In summary, a COMBINE archive is a single file that aggregates all data files and information necessary to reproduce a simulation study.

4.2.1. Implementation

The skeleton of a COMBINE archive consists of a manifest and a metadata file, specified by the Open Modeling EXchange format (OMEX). Its specification is described in a document available at identifiers.org/combine.specifications/omex.

¹[en.wikipedia.org/wiki/Deb_\(file_format\)](http://en.wikipedia.org/wiki/Deb_(file_format)), accessed 24 October 2017, see also `deb(5)` man page

²en.wikipedia.org/wiki/RPM_Package_Manager, accessed 24 October 2017, see also `rpm(8)` man page

³s.binfal.se/de/jar, accessed 24 October 2017

⁴idpf.org/epub, see also ISO/IEC TS 30135-1:2014, accessed 24 October 2017

⁵s.binfal.se/de/ecma-376, see also wikipedia.org/wiki/Office_Open_XML, accessed 24 October 2017

⁶opendocumentformat.org, specification available at www.oasis-open.org/standards, accessed 24 October 2017

Format of the COMBINE archive

The COMBINE archive is encoded using the “Open Modeling EXchange format” (OMEX). The archive itself is a “ZIP” file⁷. The ZIP format is typically used for data compression and archiving purposes. A COMBINE archive contains one or more files that can optionally be compressed to reduce the size of the archive (see Figure 4.1). The default file extension for the COMBINE archive is `.omex`. Additional extensions are available to further indicate the focus of an archive. This helps users to choose between different archives, and select appropriate software tools to open them:

- `.sedx` - SED-ML archive
- `.sbex` - SBML archive
- `.cmex` - CellML archive
- `.sbox` - SBOL archive
- `.neux` - NeuroML archive
- `.phex` - DDMoRe archive using the PharmML format

Note that a COMBINE archive may still contain files in several different standard formats. Therefore, the specific file extension is only an indication provided for the convenience of the user.

The manifest file lists the contents of a COMBINE archive

Every COMBINE archive contains at least one file, located at the root, i.e. highest in the hierarchy of files inside the archive. This mandatory file is called `manifest.xml`. It is an XML file that lists all files constituting the archive, and it describes each file’s format and location inside the archive. Source Code 4.1 shows an excerpt of a manifest file. A valid manifest file must have at least one entry representing the archive itself (first `content` entry on line 3 in Source Code 4.1). However, the manifest may contain as many entries as needed.

Each entry of the manifest file representing a file in the archive is encoded in an XML element tagged `<content>`, and characterised by three attributes. The attribute `location` contains a relative Uniform Resource Identifier (URI) [MBF05] that specifies where the file is located in the archive. In the current version of OMEX, all the files listed in the manifest must be included in the archive itself. That means, all files of a COMBINE archive need to be encapsulated within the ZIP archive. Accessing documents outside the archive might be more flexible and in concordance with a semantic web approach. However, it would require more complex verification and validation systems to answer questions such as (i) “Are the links still valid?”, (ii) “Are the versions of the files meant to be used in the archive the same as the version accessible through the URLs?”, or (iii) “How do users and software tools discriminate between network failure, non-existent file, file

⁷[wikipedia.org/wiki/Zip_\(file_format\)](https://wikipedia.org/wiki/Zip_(file_format)), technical specification: [ISO/IEC 21320-1:2015](https://iso/iec-21320-1:2015) and s.binfal.se/zip, accessed 24 October 2017

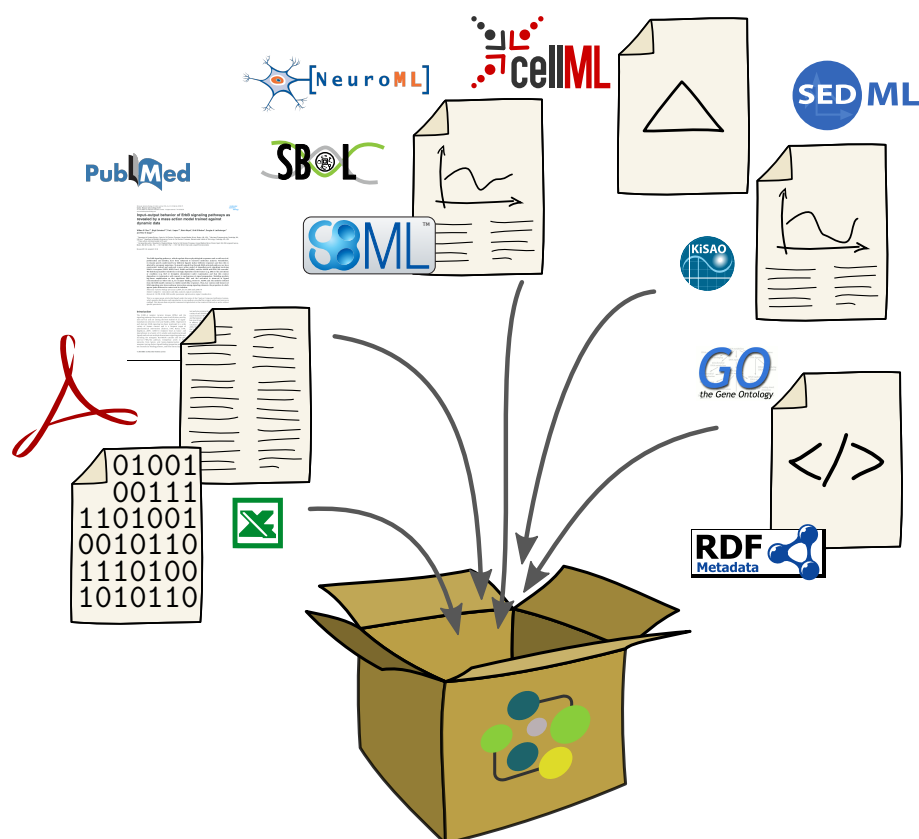


Figure 4.1. One file to share them all. A COMBINE archive is a container to ship all files relevant for a research result, such as raw data, computational models, visualisations, protocols, simulation results, semantic annotations, and metadata about its evolution. As it is built upon the ZIP format, COMBINE archives can optionally be compressed and encrypted.

in the wrong format, etc.?”. For the time being, including everything in the archive file seemed to be the safer choice. If an archive cannot be read or interpreted, the fault can be attributed to either the tool reading the archive or the tool that built the archive. It is not necessary to trust or rely on third parties.

The attribute *format* specifies the format of a file in the COMBINE archive. It should either use an Identifiers.org URI [JLL12], if it exists (e.g. for SED-ML Level 1 Version 2 <http://identifiers.org/combine.specifications/sed-ml.level-1.version-2>), or an Internet Media Type URI [FK05] (e.g. <http://purl.org/NET/mediatypes/application/pdf> for PDF files). If a format is neither registered with Identifiers.org nor with the Internet Assigned Numbers Authority (IANA), an unregistered internet media type can be used. Such unregistered media types should take the form `type/x.name` (such as <http://purl.org/NET/mediatypes/application/x.matlab> for Matlab files, or <http://purl.org/NET/mediatypes/application/x.copasi> for COPASI files).

Finally, the optional attribute *master* is a boolean flag which indicates that the corresponding files should be considered the main one. The purpose of the *master* attribute, read by the processing software once the archive is loaded and parsed, is different from the archive’s

Chapter 4. Support for shareable and reproducible simulation studies

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <omexManifest xmlns="http://identifiers.org/combine.specifications/omex-manifest">
3   <content location="." format="http://identifiers.org/combine.specifications/omex" />
4   <content location="./manifest.xml"
5     ↪ format="http://identifiers.org/combine.specifications/omex-manifest" />
6   <content location="./README.md"
7     ↪ format="http://purl.org/NET/mediatypes/text/x-markdown" />
8   <content location="./model/BIOMD0000000144.xml"
9     ↪ format="http://identifiers.org/combine.specifications/sbml.level-2.version-1" />
10  <!-- ... -->
11 </omexManifest>
```

Source Code 4.1. Excerpt of a COMBINE archive manifest. The code shows parts of the `manifest.xml` of the fully featured COMBINE archive presented in Section 4.3. Every `content` element represents a file in the archive.

extensions described above. For instance, the manifest of an archive containing a simulation description and all the models necessary to run a simulation, could have the master flag set to `true` for the SED-ML file. In an archive containing a modular model made up of many parts that are hierarchically linked, the *master* flag should be set to `true` for the main model, which in turn is importing sub-models but is not called by any.

Metadata files add semantics to a COMBINE archive

Any type of file can be included in a COMBINE archive, and therefore any type of metadata format may be used to encode clerical information and to attach semantics to the modelling contents in the archive. However, for interoperability I recommend an XML serialisation of the Resource Description Framework (RDF, [LS99; RDF14]) in the initial specification, which is stored in a file called `metadata.rdf` and reuses several existing standard terminologies:

- The Resource Description Format itself.
- vCard 4 [Per11], a standard for electronic business cards; in particular its terms `hasName`, `given-name`, `family-name`, `hasEmail`, `organization-name`. How to use vCard in RDF is specified by the vCard ontology, a recommendation from the W3C [IM14].
- Metadata terms of the Dublin Core Metadata Initiative [DCM12], in particular the terms `description`, `creator`, `created`, `modified`, `W3CDTF` (to encode a date, see [WW97]). More information on the use of Dublin Core in RDF can be found on the Dublin Core website [Nil+08].

An excerpt of a real-world `metadata.rf` is shown in Source Code 4.2 The metadata file should provide sufficient information to follow the MIRIAM and MIASE guidelines whenever possible. At the very least, the metadata file should provide the archive's creation date, the date of the last update, and details on the creator(s). COMBINE archive creators should provide a description

of the software tool that generated the archive, as well as a reference to external information describing the work. In addition, metadata about any of the archive's files may be provided. This information helps building logs of actions performed as well as audit trails, which are, for instance, essential for models used to develop drugs. Future development of the metadata content and format is subject to current discussions in the COMBINE community.

Several tools providing support for the COMBINE archive have already been released, a subset is listed in Table 4.1.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  ↪ xmlns:dcterms="http://purl.org/dc/terms/"
  ↪ xmlns:vCard="http://www.w3.org/2006/vcard/ns#">
3 <rdf:Description rdf:about="./documentation/Calzone2007.pdf">
4   <dcterms:description>Article published in Mol Syst Biol. 2007; 3: 131. DOI:
  ↪ 10.1038/msb4100171, downloaded June 11th, 2015 . It describes the models
  ↪ /model/calzone_thieffry_tyson_novak_2007.cellml and /model/BIOMD000000144.xml
  ↪ .</dcterms:description>
5   <dcterms:creator>
6     <rdf:Bag>
7       <rdf:li rdf:parseType="Resource">
8         <vCard:n rdf:parseType="Resource">
9           <vCard:family-name>Calzone</vCard:family-name>
10          <vCard:given-name>Laurence</vCard:given-name>
11        </vCard:n>
12        <vCard:org rdf:parseType="Resource">
13          <vCard:organization-name>Molecular Network Dynamics Research Group of
  ↪ Hungarian Academy of Sciences and Budapest University of Technology and
  ↪ Economics, Budapest</vCard:organization-name>
14        </vCard:org>
15      </rdf:li>
16    </rdf:Bag>
17  </dcterms:creator>
18  <dcterms:created rdf:parseType="Resource">
19    <dcterms:W3CDTF>2007-01-08T00:00:00Z</dcterms:W3CDTF>
20  </dcterms:created>
21  <!-- ... -->
22 </rdf:Description>
23 <!-- ... -->
24 </rdf:RDF>

```

Source Code 4.2. Excerpt of a COMBINE archive meta data file. The code shows parts of the `manifest.xml` of the fully featured COMBINE archive presented in Section 4.3. Every content element represents a file in the archive.

Chapter 4. Support for shareable and reproducible simulation studies

Name	Type	Link
COMBINE Archive	C# library	github.com/fbergmann/CombineArchive
LibCombine	C++ library	github.com/sbmlteam/libCombine
CombineArchive library	Java library	sems.uni-rostock.de/projects/combinearchive
CombineArchiveWeb application	Web interface	cat.bio.informatik.uni-rostock.de
JWS online	Repository and online simulator	jjj.bio.vu.nl
libCombineArchive	Java library	github.com/mglont/CombineArchive
Physiome Model Repository	Repository	models.cellml.org
pyCombineArchive	Python library	github.com/FreakyBytes/pyCombineArchive
PySCeS	Simulator	pysces.sourceforge.net
ro-combine-archive	converter	github.com/stain/ro-combine-archive
Tellurium	Simulator	tellurium.analogmachine.org
SED-ML Web Tools	Online simulator	sysbioapps.dyndns.org/SED-ML_Web_Tools
VCell	Simulator	vcell.org

Table 4.1. A collection of tools, libraries, and databases supporting COMBINE archives.

4.2.2. Use cases for COMBINE archives

Combining several types of information encoded in different formats into a single file enhancing reproducibility and fostering reuse is obviously useful for many researchers. In the following section I showcase a few applications of the COMBINE archive. These examples are not meant to be limiting, they just demonstrate the current usage of the archive.

Use case 1: Systems biology models

As recognised by the MIRIAM and MIASE guidelines, the sole description of model variables and their relationships (the structural model) is not sufficient to allow the reproduction of simulation results. The description of the simulation and analysis tasks is also necessary. Many modeling and simulation software configuration formats include both model description and experiments. In order to exchange this information in standard, tool-independent, formats, one must provide the models (for instance encoded in SBML) and the simulation description (in SED-ML). I showcase an example of such an archive in Section 4.3. Although a SED-ML file can describe an experiment using models available remotely (using URIs to allow for their identification and retrieval), it is often useful to be able to provide all the necessary information

at once. One might also need several models to reproduce a simulation experiment, for instance to compare results generated from different models, to parametrise a model using the results of another model or to run several models concurrently (see also use case 3). Moreover, a model can be described in multiple documents, for instance when encoded in CellML 1.1 [NB08] or in SBML Level 3 with the model composition package [Smi+13]. The COMBINE archive permits researchers to share all the documents describing a model and associated simulations in a single file.

Use case 2: Drug discovery models

Assessing the effects of a drug using mathematical models involves several steps, including model selection, parameter estimation, population simulations etc., both on the pharmacokinetics (drug concentration over time) and pharmacodynamics (drug effect versus drug concentration) side. The US Food and Drug Administration, for example, requires the provision of information sufficient to completely reproduce the evaluation of a drug⁸. This includes SAS transport files⁹ to represent all datasets used for model development and validation; ASCII text files of model codes or control streams and output listings for all major model building steps (base structural model, covariates models, final model, and validation model); individual plots for a representative number of subjects for population analysis; and standard model diagnostic plots. Each data item must be accompanied by a description provided in a PDF file. A structured archive containing all these components is a convenient method of ensuring that all information are faithfully transmitted, and that the correct versions of each piece of information are included in the transmission. PharmML, the emerging standard for pharmacometrics models, already supports the COMBINE archive as a container that encapsulates all the information pertaining to a pharmacometrics modelling project [Swa+15].

Use case 3: Large hybrid modular models

As systems biology moves toward the description of more complex systems, such as comprehensive biological processes [Thi+13], whole cells [Kar+12], and organs [Sch+14c], larger and more detailed models are being developed. These models encompass biological processes that might require the use of different modelling approaches. Their simulations sometimes require the use of several simulation tools. The tools' results influence each other, for instance using synchronisation [Kar+12; MN13]. To reproduce such simulation experiments, all the sub-models, all the simulation descriptions, and the descriptions of the overall experiments with the coordination of the elementary simulations must be provided. Moreover, all files must be at the correct location and in the proper version. A single file archive offers a convenient way to share consistent instances of those models.

⁸s.binfalse.de/fda, accessed 3 April 2017

⁹s.binfalse.de/ts140 and s.binfalse.de/ts140-2, accessed 14 July 2017

Use case 4: Automatic (machine-only) transfer of research results

Given a suitable technical infrastructure, no human interaction is necessary to transfer simulation jobs and simulation results between machines or applications. Indeed, a COMBINE archive may be seen as an autonomous container for all files necessary to run a simulation job on a computational model. The archive can be submitted to a compute node, which can then automatically read and process the corresponding tasks. For example, the Functional Curation project of Chaste [CMN11] uses the COMBINE archive as standard format to transfer data between the web interface and the back-end. The goal of the Functional Curation project is to compare a model's behaviour under different experimental scenarios. On the web page, a user may choose the set of computational models of interest together with a set of experiment descriptions they would like to test against these models. All selected files are compiled into a COMBINE archive, for instance using the CombineArchive library [Sch+14a], and sent to a node in a back-end, which is able to understand the encoded job and run the experiment. Afterwards, the simulation results are again packed into a COMBINE archive and sent back to the web server where they are presented to the user. Thus, the COMBINE archive eases the communication between nodes in a network. The Functional Curation project will also be subject of Section 4.6.

4.3. A fully featured COMBINE archive of a simulation study on syncytial mitotic cycles in *Drosophila* embryos

In the following sections I present a fully featured COMBINE archive, which encodes an investigation of the syncytial mitotic cycles in *Drosophila* embryos [Cal+07]. The study published by Calzone *et al.* proposes a dynamical model for the molecular events underlying rapid, synchronous, syncytial nuclear division cycles in *Drosophila* embryos. This particular study was chosen for several reasons. First, the paper, the documentation, and the related data are openly accessible. Second, the model is available in two standard formats: The CellML encoding is available from the Physiome Model Repository at s.binfal.se/pmr-calzone07 and the SBML encoding is available from BioModels Database at www.ebi.ac.uk/biomodels-main/BIOMD0000000144. Third, both model files are already curated, which increases the level of trust. Fourth, the model describes a common biological system (cell cycle). Thus, the basic mechanisms of the encoded biology should be familiar to many researchers, reducing the effort of understanding the example.

This archive contains files that are openly available for download, as well as previously unpublished files that were generated using COMBINE-compliant software tools. When executed, it reproduces the original findings by Calzone *et al.*

4.3.1. Materials and methods

The fully featured COMBINE archive was created in three subsequent steps. First, all available materials relating to the study were automatically retrieved from an online resource (initial archive). Second, the data files were organised into subdirectories, following the different aspects of a simulation study (documentation, model, experiment, result). Third, missing files were manually retrieved from web resources or created using COMBINE-compliant software tools. The three steps are described in the following.

Retrieving an initial COMBINE archive

The initial version of the COMBINE archive was generated using the web-based software tool M2CAT [SW15] Version 0.1 (m2cat.bio.informatik.uni-rostock.de). Among the suggested archives for the work by Calzone *et al.*, I chose the simulation study containing a CellML model and a visualisation of the model in three different formats (PNG, SVG, AI). M2CAT automatically generated the initial COMBINE archive from these files. It also added metadata to the archive, such as annotations to creators, contributors, and modification times. M2CAT retrieved this metadata from the corresponding Git project in the Physiome Model Repository (`git log`).

Organising the COMBINE archive

For convenience, the files inside the COMBINE archive were structured in subfolders. The initial archive was therefor imported into the CombineArchiveWeb application (WebCAT, [Sch+14b], see also Section 4.4.2) Version 0.4.13 (cat.bio.informatik.uni-rostock.de). WebCAT is a web interface to display and modify the files contained in an archive, together with metadata and file structures. The files inside the archive were organised in four directories, which reflect the different aspects of a simulation study:

- `documentation/`: Files that describe and document the model and/or experiment (*empty*)
- `model/`: Files that encode and visualise the biological system (*4 files*)
- `experiment/`: Files that encode the *in silico* setup of the experiment (*empty*)
- `result/`: Files that result from running the experiment (*empty*)

All files in the initial archive were stored in the `model/` directory. However, these files alone are not sufficient to reproduce the study.

Extending the COMBINE archive

To render the encoded study reproducible, the COMBINE archive needs to be extended with additional files.

Chapter 4. Support for shareable and reproducible simulation studies

The article is typically the central object of a research study. For this study, the original publication by Calzone *et al.*, together with available supplementary information, was retrieved from the homepage of the journal *Molecular Systems Biology* (msb.embopress.org/content/3/1/131). Using WebCAT, the files were uploaded to the `documentation/` directory of the archive. The automatically added metadata was adjusted to attribute the authors of the publication and to state when and where the files were downloaded. In the background, WebCAT encoded the metadata in RDF/XML and added it to the archive.

The model is also available in SBML format. It was retrieved from BioModels Database (www.ebi.ac.uk/biomodels-main/download?mid=BIOMD0000000144, SBML Level 2 Version 1) and uploaded to the `model/` directory. Again, the metadata was corrected to attribute the original authors, curators, and contributors, as stated on the BioModels Database website (www.ebi.ac.uk/biomodels-main/BIOMD0000000144) and in the model document.

The simulation description is essential to run the experiment. It defines the simulation environment and the output of the *in silico* execution. As no simulation description was found in any of the open repositories, an initial version was created using the SED-ML Web Tools (SWT) Version 2.1 (sysbioapps.dyndns.org/SED-ML_Web_Tools). SWT takes the model files and creates a default simulation description with standard settings. For this study, a default SED-ML file encodes instructions to generate 66 plots and a data table. Each plot describes the change of concentration in one species of the model. The data table contains all numerical values. Based on the default script, a second SED-ML file (`Calzone2007-simulation-figure-1B.xml`) was generated to reassemble Figure 1B of the original publication. Using WebCAT, both SED-ML scripts were added to the `experiment/` directory of the archive and the metadata was recorded.

The simulation results reflect the behaviour of a model under certain conditions. The script defined in `Calzone2007-simulation-figure-1B.xml` was loaded into SWT and into the stand-alone software program COPASI Version 4.15 Build 95 [Hoo+06]. The plots generated by both tools show that the *in silico* experiment reproduces the results from the paper. Using WebCAT, the figures produced by SWT and COPASI were uploaded and added to the `result/` directory of the archive. Metadata, such as the versions of the software tools, was added accordingly.

The visualisation of a model helps to understand the encoded biological system. For this study, an SBGN-compliant visualisation of the model was created using SBGN-ED Version 1.5.1 [CKS10] together with VANTED Version 2.1.0 [Roh+12]. The automatic layout generated by SBGN-ED was then improved manually in multiple iterations. The resulting Figure 4.2 was exported in different formats (GraphML [Bra+01a], GML¹⁰, PNG, PDF, and SBGN-ML [Jer+12]) and uploaded to WebCAT's `model/sbgn` directory together with metadata.

¹⁰www.fim.uni-passau.de/index.php?id=17297&L=1, accessed 16 July 2017

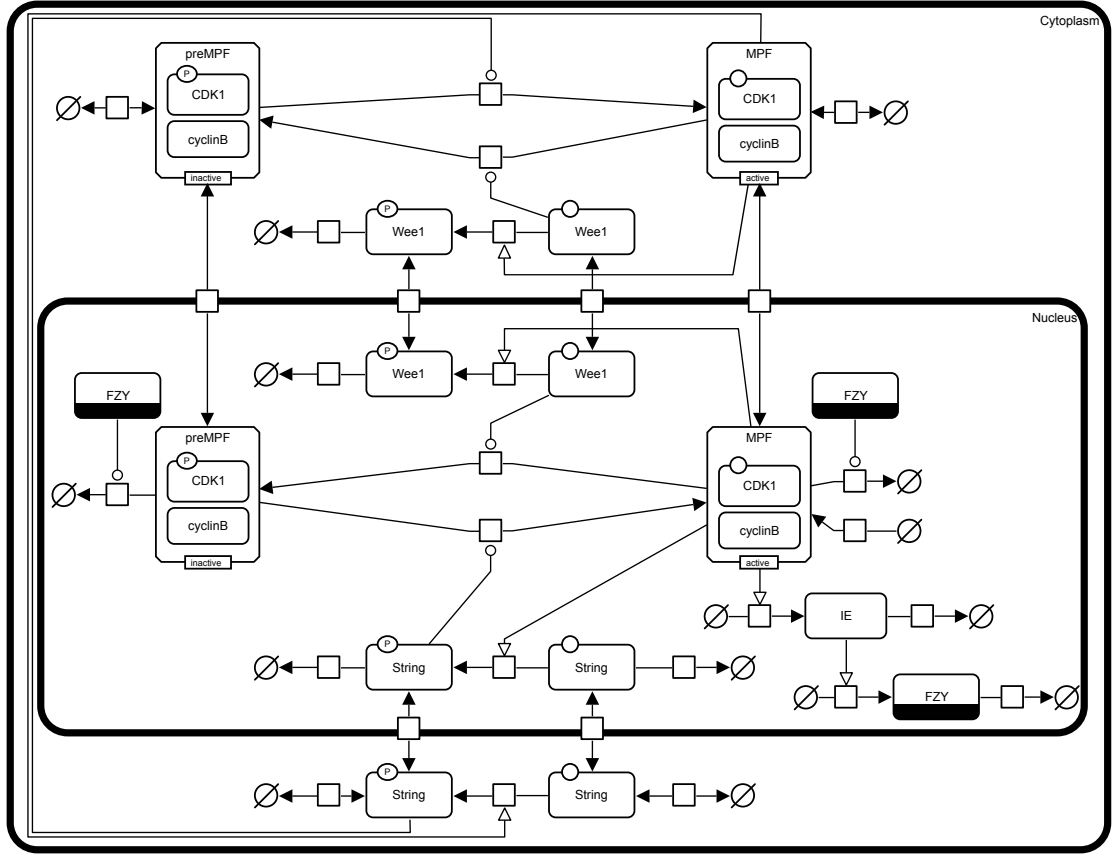


Figure 4.2. Visualisation of the model. This figure shows the SBGN-PD compliant reaction network, as encoded in the SBML model obtained from BioModels Database. The figure was generated and modified using SBGN-ED.

4.3.2. Data description

The archive consists of 21 files, see Table 4.2. Among these files are the `manifest.xml` (see excerpt in Source Code 4.1) and the `metadata.rdf` (see excerpt in Source Code 4.2), which found the skeleton of the archive. The manifest lists the files included in the archive. The metadata file contains additional information about the files in the archive, such as creators and descriptions. A third file, `README.md`, contains a description for visitors of the GitHub repository, where the archive is being developed (github.com/SemsProject/CombineArchiveShowCase). The remaining 18 files are organised in four directories, compare Section 4.3.1. The original publication (PDF) is stored in the `documentation/` directory. The encodings of the model (CellML, SBML, graph formats) are stored in the `model/` directory. The simulation descriptions (SED-ML) are stored in the `experiment/` directory. The simulation results (SVG, PNG) are stored in the `result/` directory.

The latest version of the compiled COMBINE archive can be accessed through our web server at scripts.bio.informatik.uni-rostock.de/getshowcase.php.

Chapter 4. Support for shareable and reproducible simulation studies

File	Format	Description
manifest.xml	Omex	Skeleton, automatically generated by WebCAT
metadata.rdf	Omex	Skeleton, automatically generated by WebCAT
README.md	Markdown	Human-readable information for users stumbling upon the archive
model/		
BIOMD0000000144.xml	SBML	origin: BioModels Database
calzone_2007.svg	SVG	origin: Physiome Model Repository
calzone_2007.ai	Illustrator	origin: Physiome Model Repository
calzone_2007.png	PNG	origin: Physiome Model Repository
calzone_thieffry_tyson_novak_2007.cellml	CellML	origin: Physiome Model Repository
sbgn/Calzone2007.gml	GML	SBGN compliant fig. generated using SBGN-ED
sbgn/Calzone2007.graphml	GraphML	SBGN compliant fig. generated using SBGN-ED
sbgn/Calzone2007.pdf	PDF	SBGN compliant fig. generated using SBGN-ED
sbgn/Calzone2007.png	PNG	SBGN compliant fig. generated using SBGN-ED
sbgn/Calzone2007.sbgn	SBGN-ML	SBGN-ML encoding exported with SBGN-ED
experiment/		
Calzone2007-default-simulation.xml	SED-ML	Simulation description generated using SED-ML Web Tools
Calzone2007-simulation-figure-1B.xml	SED-ML	Simulation description generated using SED-ML Web Tools based on Calzone2007-default-simulation.xml
documentation/		
Calzone2007.pdf	PDF	Scientific publication “ <i>Dynamical modeling of syncytial mitotic cycles in Drosophila embryos</i> ” obtained from msb.embopress.org/content/3/1/131
Calzone2007-supplementary-material.pdf	PDF	Supplementary information for the publication obtained from msb.embopress.org/content/3/1/131
result/		
Fig1B-bottom-COPASI.svg	SVG	Image generated by executing Calzone2007-simulation-figure-1B.xml on BIOMD0000000144.xml in COPASI
Fig1B-top-COPASI.svg	SVG	Image generated by executing Calzone2007-simulation-figure-1B.xml on BIOMD0000000144.xml in COPASI
Fig1B-bottom-webtools.png	PNG	Image generated by executing Calzone2007-simulation-figure-1B.xml on BIOMD0000000144.xml in SED-ML Web Tools
Fig1B-top-webtools.png	PNG	Image generated by executing Calzone2007-simulation-figure-1B.xml on BIOMD0000000144.xml in SED-ML Web Tools

Table 4.2. Content of the fully featured COMBINE archive. The table lists all files included in the presented COMBINE archive together with formats and descriptions. The indentation in the *File* column indicates the directory structure used to organise the files in the archive. The origin *BioModels Database* refers to www.ebi.ac.uk/biomodels-main/BIOMD0000000144, and the origin *Physiome Model Repository* refers to models.cellml.org/workspace/calzone_thieffry_tyson_novak_2007. The visualisation in `model/sbgn/` is shown in Figure 4.2.

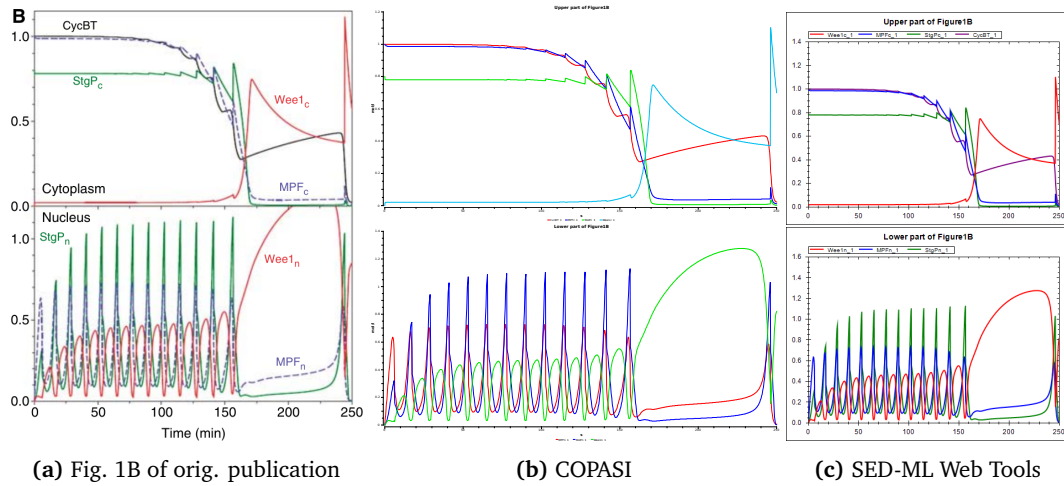


Figure 4.3. Comparison of simulation results. The figure shows the simulation results included in the original publication (4.3a). Furthermore, the results generated by COPASI (4.3b) and the SWT (4.3c), using the SED-ML script `Calzone2007-simulation-figure-1B.xml`, are shown. Even though the legends may not be readable, they are retained deliberately to indicate the different tools.

4.3.3. Data validation and conclusion

The presented COMBINE archive provides a reproducible simulation study for a previously published model on syncytial mitotic cycles in *Drosophila* embryos [Cal+07]. The archive contains several files that were collected from online resources, e. g. the CellML model from the Physiome Model Repository or the scientific publication from the publisher’s website. It also provides new files that did not exist previously, e. g. a SED-ML file to encode the simulation setup for Figure 1B of the original publication.

Together, this fully featured archive allows scientists to reproduce the results obtained by Calzone *et al.* in software tools that support COMBINE archives. To validate the reproducibility, the archive was executed in three different simulation tools. For example, I ran the encoded simulation study in COPASI, see Figure 4.3b. I also loaded the archive to the SWT by opening a specific URL (sysbioapps.dyndns.org/SED-ML_Web_Tools/Home/SimulateUrl?url=https://scripts.bio.informatik.uni-rostock.de/getshowcase.php). Clicking the URL leads immediately to the visualisation of the simulation results in the web browser, see Figure 4.3c. Moreover, users reported a successful reproduction of the original simulation results using Tellurium [Sau+16] (github.com/SemsProject/CombineArchiveShowCase/pull/2).

This section describes the fully featured COMBINE archive as published on Figshare [ST16]. However, I expect the archive to evolve further. The latest version of the archive is available from GitHub at github.com/SemsProject/CombineArchiveShowCase (latest commit at the time of writing: 0e501db). It can also be downloaded from our website at scripts.bio.informatik.uni-rostock.de/getshowcase.php. Extensions, refinements, and comments are very welcome. Please fork the project on GitHub and contribute pull requests.

4.4. Tool support for COMBINE archives

To increase the usability of the OMEX format, I (co-)developed a number of software tools for developers and modellers who wish to create, read, and modify COMBINE archives. The first implementation provides a Java library which supports full functionality as defined in the OMEX specification (Section 4.4.1). The second implementation, available via a web portal, allows users to explore and share COMBINE archives on the internet (Section 4.4.2). The third implementation permits online simulation of COMBINE archives (Section 4.4.3). The fourth implementation is a method to search for COMBINE archives, which also integrates previously mentioned tools. It will be discussed in Section 4.5 in detail. Together, the tools provide means to (1) create archives, e.g. from a modeling project to be submitted to a journal or repository, (2) modify an existing archive, e.g., optimising the algorithm in the simulation setup, (3) explore other people's work, e.g. by simulating an obtained archive on a web-based platform, and (4) share archives with collaborators without bothering about single files or formats (see Figure 4.4).

4.4.1. The CombineArchive library

The COMBINE archive library¹¹ represents the core of most of the following tools. It implements the latest COMBINE archive specification and, thus, breaths life into the idea of sharing complete containers of files necessary to reproduce an *in silico* experiment. Operating directly on a zip file system, the library offers all necessary methods to create, explore, and modify COMBINE archives, including:

- Extracting single files or the whole archive
- Browsing through the archive
- Adding and removing files
- Renaming and reorganising files
- Attaching and retrieving meta information

Moreover, the CombineArchive library tries to parse and translate the meta information into an internal representation. Currently, I just support the metadata format as defined in the OMEX specification. However, the parser can be extended with further strategies to understand and interpret any other RDF/XML representation. Given this layer of abstraction, other tools can implement support for COMBINE archives without bothering about manifest files or metadata associated to archive entries. The CombineArchive library was created for tool developers. It has already been integrated with software such as the the CombineArchiveWeb application, the Functional Curation Project, and M2CAT. With multiple applications at hand, there is plenty of example code available on how to use the library.

¹¹semsproject.github.io/CombineArchive, accessed 24 October 2017

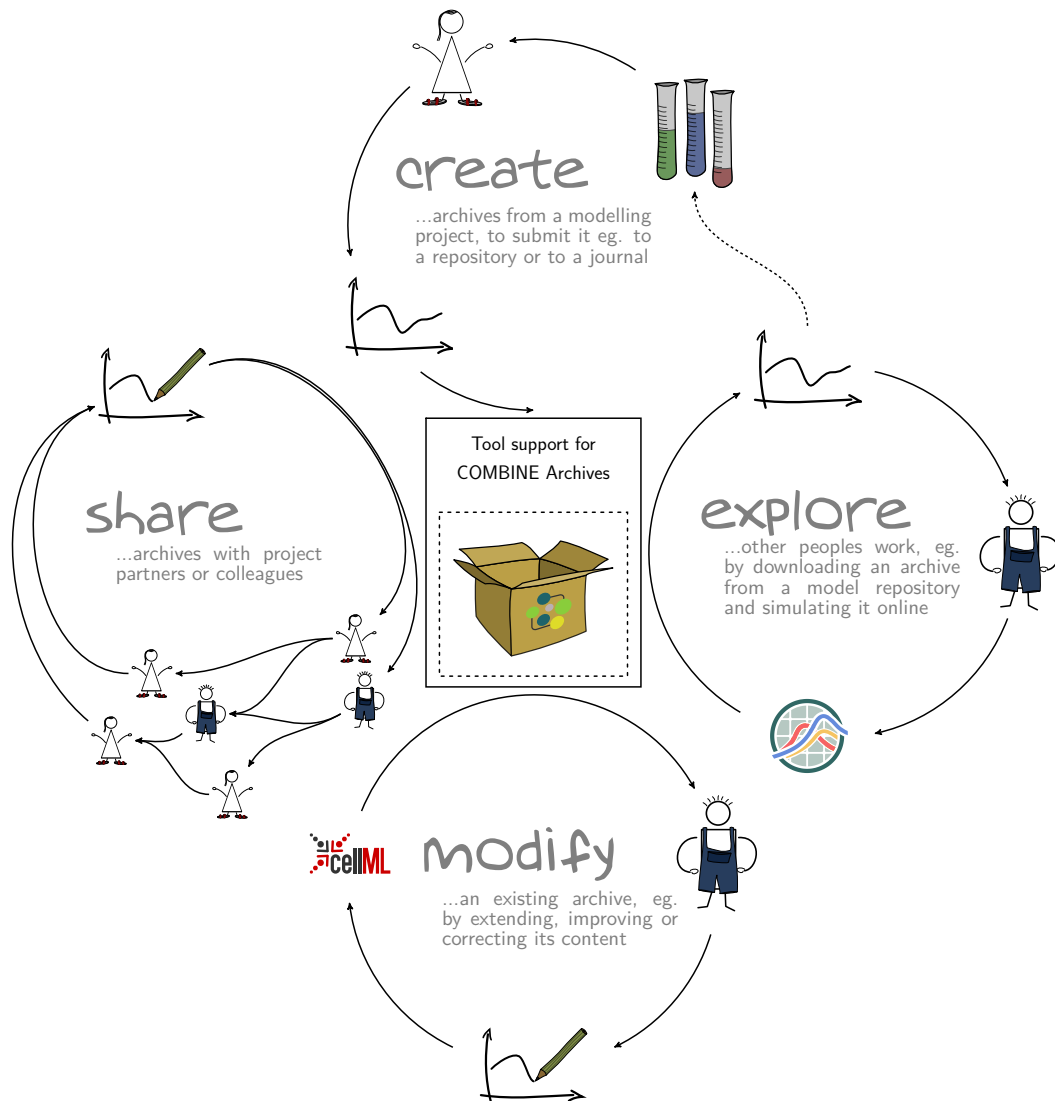


Figure 4.4. Visual motivation for COMBINE archives. Using, for example, the tools listed in Table 4.1 it is possible to create and modify COMBINE archives. Appropriate tool support helps researchers to share and explore reproducible research results, which ultimately fosters model reuse.

4.4.2. The CombineArchiveWeb application

The CombineArchiveWeb application¹² uses the CombineArchive library as a code base. It enables researchers to work with the COMBINE archive format on the internet. Additionally, the application offers RESTful services¹³, which can be used by third party client applications. User with special interest in privacy do not need to alienate their research. Since the CombineArchiveWeb application is openly available and easy to install¹⁴ (e.g. using the provided Docker

¹²cat.bio.informatik.uni-rostock.de, accessed 14 July 2017

¹³semsproject.github.io/CombineArchiveWeb/Api, accessed 14 July 2017

¹⁴semsproject.github.io/CombineArchiveWeb/BuildAndInstall, accessed 14 July 2017

Chapter 4. Support for shareable and reproducible simulation studies

container¹⁵), anyone can host their own instances on private servers. In order to prohibit abuse, the maintainer of an installation can easily configure quotas for the users. For example, it is possible to limit the maximum file size, the maximum age of an untouched archive, the number of archives per workspace, or the number of files in an archive. Since the CombineArchiveWeb application is not intended as a database for long-term storage of simulation studies old archives are deleted regularly.

Researchers who wish to share their results through the CombineArchiveWeb application can simply upload all relevant files belonging to a piece of modelling work. This automatically creates a workspace. Workspaces are an ideal tool to manage files. They foster collaborations and prevent inconsistencies in versions of files. Uploaded files are immediately bundled in a COMBINE archive, which can be downloaded at any time and from any location. The created workspace can also be shared with collaborators to work from different physical locations. The CombineArchiveWeb application supports researchers in exploring and reproducing scientific results. For example, authors of a publication may decide to provide their code as a COMBINE archive. When users open the archive, the CombineArchiveWeb application automatically reads the files and their metadata. It builds the links between the various files and presents the contents in a human-readable format (see Figure 4.5). Using the API of the SED-ML Web Tools, which will be subject of Section 4.4.3, it is possible to automatically execute a simulation encoded in the COMBINE archive.

The CombineArchiveWeb interface integrates a data management strategy including a number of benefits, such as the easy exchange of data, the independence from the system a users is working on, and the possibility to collaborate with other scientists. A centralised data storage allows users to work on the same data from different workplaces. Moreover, it is possible to share a workspace with collaborators through an HTTP-Link. Even if this feature is not ready for real-time collaboration, it potentially increases the productivity of modellers. Furthermore, the web interface provides connections to model databases, such as the Physiome Model Repository. So it is possible to create COMBINE archives directly from repositories hosted at the Physiome Model Repository, as well as any other repository based on either Mercurial¹⁶ or Git¹⁷ source code management system (including e.g. GitHub¹⁸). COMBINE archives can be organised in work spaces. Files in an archive are accessible through a web-based file browser. Metadata associated with a selected file are presented in a human-readable format, including annotations of the specific file type according to the COMBINE archive specification as well as proprietary annotations in RDF/XML format. Master files, i.e. files the creator of the COMBINE archive deemed most important, are emphasised. The archives and workspaces can be shared with others through simple HTTP links.

¹⁵hub.docker.com/r/binfalse/webcat, accessed 14 July 2017

¹⁶mercurial-scm.org, accessed 14 July 2017

¹⁷git-scm.com, accessed 14 July 2017

¹⁸github.com, accessed 14 July 2017

CombineArchiveWeb

the current workspace contains the following archives:

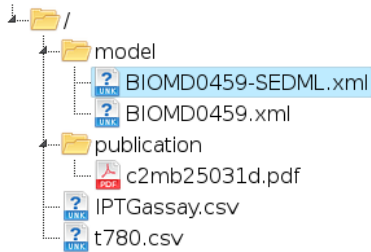
::liebal2012_cat

[\[start\]](#) [\[about\]](#) [\[create\]](#)

id: cc54b7d5-2815-40c4-89c9-5a8d3c0b1dcb
name: liebal2012_cat
[\[Download\]](#) [\[Edit\]](#) [\[Delete\]](#)

UPLOAD FILES

Archive Content



BIOMD0459 - SEDML.xml

file name: BIOMD0459-SEDML.xml
file path: /model/BIOMD0459-SEDML.xml
format: <http://purl.org/NET/mediatypes/application/xml>
size: 3.76 KB
master: yes

[\[Add OMEX meta\]](#) [\[Add XML:RDF meta\]](#) [\[Download\]](#) [\[Edit\]](#) [\[Delete\]](#)

OMEX entry

created: 10/20/2014, 1:48:24 PM
modified: [10/20/2014, 1:48:24 PM]
[11/16/2014, 9:39:56 PM]
description: The simulation description to reproduce the results as reported in Mol.BioSyst 2012.
creators:
Ulf Liebal *University of Rostock*
ulf.liebal@uni-rostock.de

[\[Edit\]](#) [\[Delete\]](#)

Figure 4.5. Screenshot of the CombineArchiveWeb application. The web interface provides means to: (i) create archives, e.g. from a modelling project to be submitted to a journal or an open repository; (ii) explore other people's work, e.g. by downloading archives from the Physiome Model Repository and studying included files; (iii) modify an existing archive, e.g. by improving, extending, and correcting its content; and (iv) share archives with project partners. The figure shows a workspace containing a single COMBINE archive (liebal2012_cat) with the identifier cc54b7d5-2815-40c4-89c9-5a8d3c0b1dcb. The archive's content is shown at the bottom of the page. A directory tree lists the composition of files (left-hand side), and their metadata as recorded in the COMBINE archive (right-hand side). Single files can be downloaded and deleted, and their metadata can be modified. Using the upload button at the top-right of the web page, new files can be added to the archive.



Figure 4.6. SED-ML Web Tools importing and then simulating a fully featured COMBINE archive [ST16]. To reproduce, just click sysbioapps.dyndns.org/SED-ML_Web_Tools/Home/SimulateUrl?url=http://ndownloader.figshare.com/files/5370005

4.4.3. Generation and execution of reproducible simulation studies using the SED-ML Web Tools

We developed the SED-ML Web Tools to support users in generating, modifying, simulating, and exporting standard-compliant simulation experiments [Ber+17]. The Web Tools are implemented in ASP.NET MVC¹⁹ and provide a web-based interface to simulation studies using the SED-ML format. They are based on libSedML²⁰, a .NET library that is also available independently.

The SED-ML Web Tools provide an easy-to-use wizard to generate SED-ML files for a model. The model can be specified by (i) uploading it to the portal, (ii) providing a link to a web server delivering the model file, or (iii) providing a unique resource identifier (URN). Once the model is obtained, the SED-ML Web Tools scan the model's structure for parameters, derive initial values and generate a working SED-ML file using a default configuration (e.g., for a time course simulation). This wizard, however, is at the moment of writing only available for SBML encoded models.

¹⁹www.asp.net/mvc, accessed 14 July 2017

²⁰libsedml.sf.net, accessed 14 July 2017

An integrated editor allows for revising and modifying the simulation description, e. g. to study an alternative behaviour of the system or to generate plots for different parametrisations. It is available for models encoded in either the CellML or SBML format. The editor supports the SED-ML XML format and a Python-based Script Language [Ber11].

Using libSedML, SED-ML files can immediately be simulated on the server and the results are presented in the web browser (Figure 4.6). The SED-ML Web Tools run SBML models using RoadRunner [BS06], Gillespie, and LPsolve. CellML models are run using CSim²¹. Simulation studies can be exported as (i) standalone SED-ML descriptions; (ii) SED-ML archives; COMBINE archives (iii) with or (iv) without simulation results. The Web Tools are also able to read and understand simulation studies encoded in COMBINE archives. COMBINE archives are extracted and the encoded simulations are run immediately.

The SED-ML Web Tools offer a sophisticated API, which can easily be integrated with other tools. For example, the CombineArchiveWeb links to the SED-ML Web Tools to realise instantaneous simulation of studies. A single click on a link launches the SED-ML Web Tools, triggers the download of the corresponding archive from the CombineArchiveWeb's web server, and executes the simulation study encoded in the COMBINE archive. Using the API, simulation studies may even be executed without any human interaction. For example, the M2CAT tool, which is subject of the following Section 4.5, enriches studies and adds simulation results generated using the SED-ML Web Tools. A detailed documentation of the Web Tools' API²² demonstrates how to use the Web Tools within your application.

The reuse of simulation studies is essential in collaborative and responsible research. The SED-ML format and the COMBINE archive are two pioneering approaches to exchange simulation experiments in computational biology and beyond. Equipped with the SED-ML Web Tools, users get fundamental support in developing reproducible simulation studies. The API of the SED-ML Web Tools is accessible for external services, for example, to generate simulation descriptions and COMBINE archives remotely, or to execute simulation studies online.

4.5. A method to extract reproducible simulation studies from open model repositories

In this chapter I already introduced a number of tools supporting COMBINE archives. However, it still remains a challenge to find and export reproducible simulation studies. In this section, I show how the data retrieved from a database can be bundled as COMBINE archives. My prototype implementation showcases a workflow that combines two previously developed software tools: A graph database for linked storage of model-related data (Masymos, compare Section 3.3) and a web-based tool to generate and manage COMBINE archives (Combine-

²¹get.readthedocs.io, accessed 14 July 2017

²²s.binfalse.de/webtools-api, accessed 16 July 2017

Chapter 4. Support for shareable and reproducible simulation studies

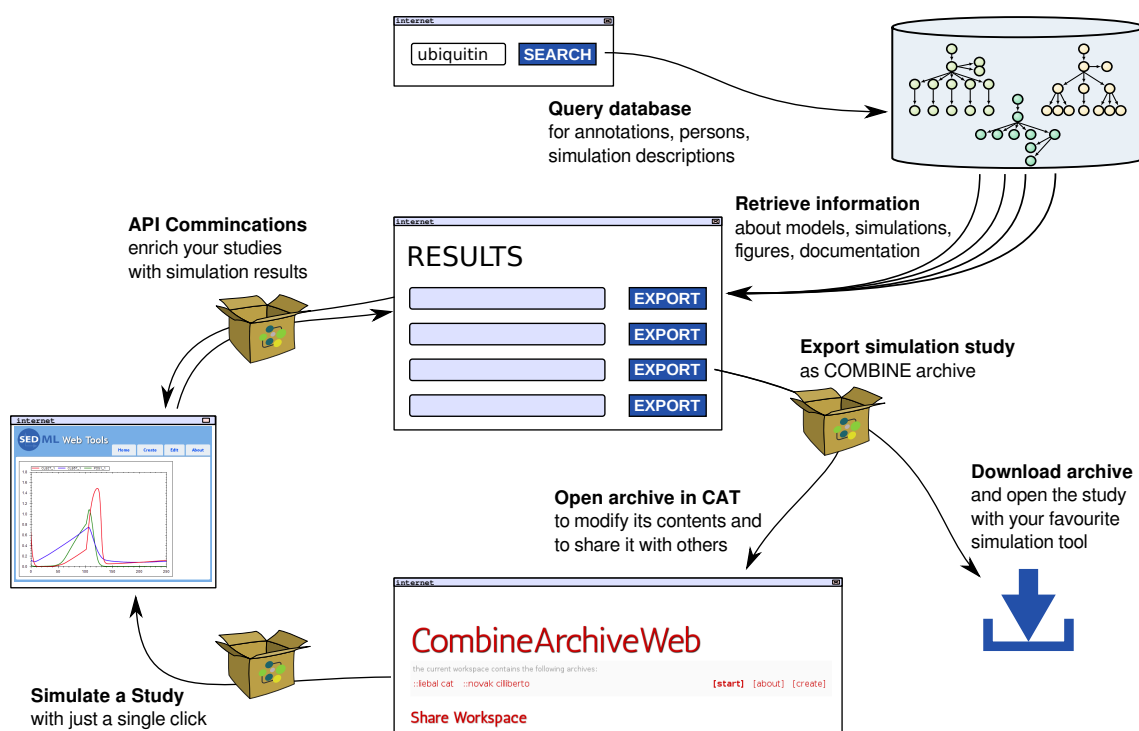


Figure 4.7. The M2CAT workflow. Keywords are used to search for model files, simulation setups, and related data in Masymos. The search results are presented in a web interface. They are organised by model name, and displayed together with the additional resources. By clicking the *export* button, users can either download the COMBINE archive of an entry, or open the archive directly in the CombineArchiveWeb tool. In addition, archives can be simulated using the SED-ML Web Tools.

ArchiveWeb, compare Section 4.4.2). Using M2CAT (Masymos to CAT) a model can easily be retrieved together with all other files that are necessary to understand the model and to reproduce the original results.

4.5.1. Implementation

My software M2CAT enables researchers to retrieve models or simulation studies from Masymos and directly generates COMBINE archives from all relevant files (Figure 4.7). The data can then be either downloaded or opened in the CombineArchiveWeb interface.

The workflow is implemented in a web interface at m2cat.bio.informatik.uni-rostock.de. Masymos can be queried via a simple search field. In the initial version, keywords are translated into Cypher queries and sent to Masymos via Neo4j's RESTful service. First, M2CAT consults the annotation index, which contains all terms occurring in the semantic annotations of models (Source Code 4.3). The result is a list of models and associated documents. Afterwards, M2CAT consults Masymos' person nodes, which contain the names of all authors of reference publications, curators, and other contributors of model code (Source Code 4.4). The result is again a list of models and associated documents. Finally, M2CAT merges both lists and searches

```

1 START res=node:annotationIndex("RESOURCETEXT:(ubiquitin)")
2 MATCH res<-[:is]-(:ANNOTATION)-->(s)-[:BELONGS_TO]->(m:MODEL)-[:BELONGS_TO]->(d:DOCUMENT)
3 RETURN distinct(m),d

```

Source Code 4.3. Query Masymos' annotation index. Return models *m* which contain elements annotated with *ubiquitin* and associated documents *d*.

```

1 MATCH (d:DOCUMENT)-[HAS_MODEL]->(l:MODEL)-[HAS_ANNOTATION]->
2 (k:ANNOTATION)-[HAS_PUBLICATION]->(m:PUBLICATION)-[HAS_AUTHOR]->
3 (n:PERSON{FAMILYNAME:"ubiquitin"})
4 RETURN distinct(m),d

```

Source Code 4.4. Query publications stored in Masymos. Return models *m* which are annotated with publications written by *ubiquitin* and associated documents *d*.

```

1 START known=node(273)
2 MATCH (known)-->(:MODEL)<--(:SEDML_MODELREFERENCE)<--(s:SEDML)<--(d:DOCUMENT)
3 RETURN s,d

```

Source Code 4.5. Query simulation descriptions available in Masymos. For the document with id 273: Return simulation descriptions *s* and associated documents *d*.

for additional resources associated to each entry in the merged list. For example, Source Code 4.5 retrieves all simulation descriptions that are linked to a certain model document.

The sets of documents belonging to the simulation studies are then grouped and presented on the website. For each study M2CAT shows the model name and lists available resources.

Even though, the M2CAT tool was re-implemented to use advanced search and retrieval techniques, the idea behind M2CAT is still the same. The re-implementation was done in the scope of a student's project, which I supervised. The new M2CAT version now also integrates resources from external databases, such as PubMed²³ [NCB16] or BioModels Database.

4.5.2. Integration with other tools

Using M2CAT it is now possible to export simulation studies as COMBINE archives. Masymos does not store the files themselves, but provides links to all necessary files through the retrieved document nodes. M2CAT resolves these links, retrieves the files, and utilises the CombineArchive library to create the archive. Specifically, the files are packed, the archive and its files are annotated with metadata about the contents and the creator, and the manifest file is written. The *in silico* studies can optionally be run using the SED-ML Web Tools and simulation results can be included in the archive. The user can then either download the archive or open it in the CombineArchiveWeb application (see again Figure 4.7). Downloaded archives simplify

²³www.ncbi.nlm.nih.gov/pubmed, accessed 14 July 2017

Chapter 4. Support for shareable and reproducible simulation studies

journal submissions, if the model code is required together with the manuscript. Opening the COMBINE archive in the CombineArchiveWeb application enables modellers to complete or extend the simulation study by adding further files to the archive, such as additional figures or supplemental descriptions. Further use cases of the COMBINE archive include easy sharing of files among collaborators, encapsulating datasets used for model development and validation, sharing consistent instances of a model, enabling automatic (machine-only) transfer of research results [Ber+14].

4.5.3. Summary

The systems biology community has identified reusability of simulation studies as one major challenge in the field. I presented a workflow that combines existing tools for model management and provides a user-friendly interface for downloading reproducible simulation studies. The queried database Masymos already integrates several resources for model-related data. M2CAT gathers this data and generates COMBINE archives from it. The instance of Masymos currently contains models in SBML and CellML formats and associated simulation experiments in SED-ML format. I am convinced that the usefulness of the COMBINE archives generated by M2CAT increases, if more model-related data are included. Specifically, I would like to see graphical representations of models in SBGN format and simulation results included in Masymos. However, the described workflow is applicable to model repositories in general. It reveals its full power, if the model-related data are already linked, as demonstrated by Masymos.

4.6. Functional curation of simulation studies

Mathematical and computational modelling of cardiac electrophysiology has an impressive history [Nob60; NR01]. Encoding hypotheses about how systems work in a quantitative form has yielded valuable insights into cellular behaviour and the roles of different ionic currents [Nob11], the mechanisms behind arrhythmias [Def+14; Qu+14], and treatments such as defibrillation [Tra11]. Typically for mathematical modelling, models are developed to represent specific quantitative hypotheses and to answer concrete scientific questions. Therefore, studies published about new models display behaviour under particular experimental conditions and draw inferences from that behaviour. This is, of course, appropriate and useful.

However, this success entails a challenge: How do we assess and compare the underlying hypotheses and emergent behaviours so that we can choose a model as a suitable basis for a new study or to characterise how a particular model behaves in different scenarios? If we consider that a mathematical model is a quantitatively encoded hypothesis (or set of hypotheses), how can we see which hypothesis is best supported by new data? A distinct group of researchers may be able to compare how their own models behave in a wide range of different situations [OR12], or easily vary their simulations to represent different experimental scenarios. Nevertheless,

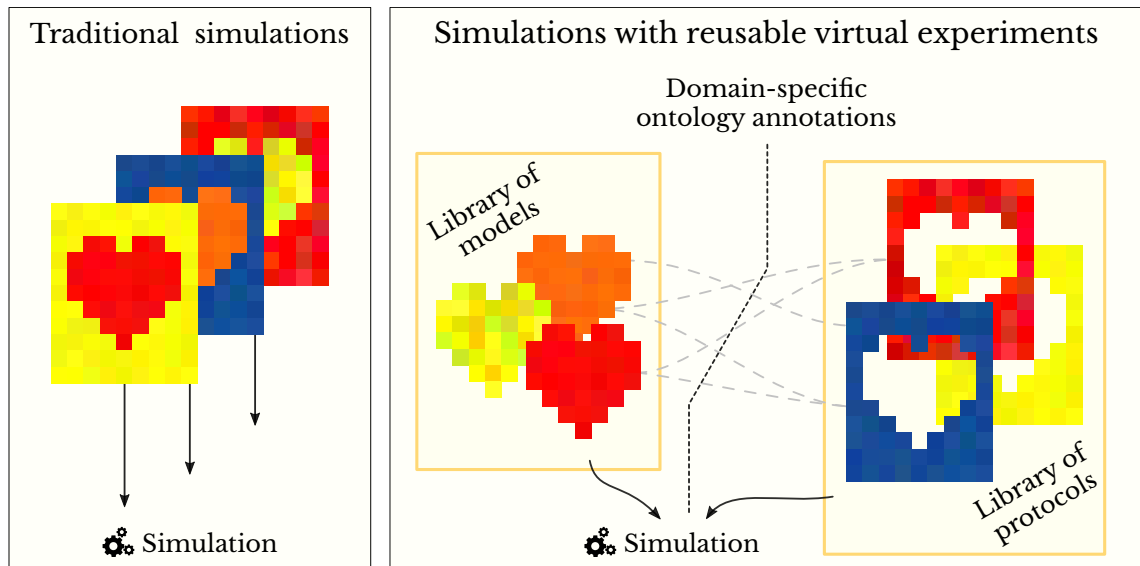


Figure 4.8. The concept of reusable virtual experiments. Traditionally, the models and protocol are strictly tied together and can usually only be simulated in that fixed combination. Instead, simulation studies should be decomposed into models and protocols. A model can then be run and analysed using different protocols and, vice versa, different models can be benchmarked using a fixed protocol.

there is no automated solution for examining how a particular model behaves under a range of experimental conditions, let alone for comparing the behaviours of any of the published models. As a result of this technical barrier, only a very few published studies have compared models and hypotheses (with rare exceptions, e.g. [CF07; Cla+11; Nie+09; Sob09; Ten+06]). The need will become particularly acute as models begin to be used in simulation studies for applications such as drug safety testing, which rely on behavioural predictions beyond the normal regime in which many models were originally developed and tested [Mir+11; Mir+12; OR12; Sag+14].

An outstanding effort has been made to encode many of the action potential (AP) models in the CellML format [Gar+08; LHN04; Llo+08]. However, despite more than 50 years of cardiac modelling, and now the availability of hundreds of models and variants for cardiac electrophysiology, investigators have had nowhere to look up simple model characteristics such as the AP waveform at a given pacing rate. There has been no automatic mechanism for checking even the published behaviours ascribed to a model, let alone other potential or expected capabilities. Given this, it is unsurprising that occasionally the curated model descriptions do not match the original implementations, and I give one example of this further below.

I therefore encourage the concept of virtual experiments, the *in silico* analogues of wet lab experiments [CVW15]. Virtual experiments are defined by protocols that can be encoded in a form amenable to processing by a computer program, and applied to different models of a system, see Figure 4.8. This could be seen as an analogue of an experimental protocol

Chapter 4. Support for shareable and reproducible simulation studies

that different labs could follow to reproduce research findings, which is increasingly being recognised as essential for experimental research [BI15]. Tools implementing this concept could address some of the challenges described above [CMN11].

Here, I present the Cardiac Electrophysiology Web Lab²⁴, a platform that realises virtual experiments. It is a user-friendly web interface that allows modellers to characterise their (and others') cardiac electrophysiology models and to compare a model's behaviour against that of any other model under a set of simulated experimental conditions. It must be emphasised that the Web Lab *does not* make any judgements as to whether the models behave appropriately in a given experiment, or which is best. Instead, it provides a system to enable careful comparison and analysis of the behaviour of models in multiple virtual experiments. The following methods section describes technical details; afterwards I showcase some of the results that are already available online to provide an impression of the potential uses, capability, and flexibility of the Web Lab.

4.6.1. Materials and methods

To automatically characterise and compare the behaviour of models in different experimental scenarios, both the models and the protocols must be described in formats that can be understood by the software, rather than as black-box programs in their own right. In addition, the details of the protocol need to be separated from the model equations, thus moving us from models of a particular experiment to models of a biological system. Different protocols may then be applied to each model of the system, exercising them in different ways. This approach is shown schematically in Figure 4.8.

Model descriptions are encoded in CellML format. The Web Lab can simulate any model that mathematically is a system of ordinary differential equations (including the trivial case of a purely algebraic model). Although we are contributing to the development of a community standard format for protocol descriptions (i.e. SED-ML), it does not yet meet all requirements. In the interim, an extended version of this language was used [CMN11], with a text syntax that facilitates understanding and editing of the protocols [CO13; Coo+11]. The main features are the use of annotations indicating the physiological meaning of model variables, to avoid confusion over naming (these can be added to the CellML files within minutes in the Web Lab), and automated unit conversions to ensure mathematically consistent simulations. These simulation tools are built on top of the Chaste libraries for computational biology [Mir+13; PG14; Pit+09].

The tools are made available to the user via a web interface to provide an installation-free, interactive experience. Although anonymous users may browse and compare these stored results, it is also possible to sign up for an account and receive permission to add your own models and protocol descriptions; these may be kept private or published for all to see.

²⁴travis.cs.ox.ac.uk/FunctionalCuration, accessed 16 July 2017

Chapter 4. Support for shareable and reproducible simulation studies

Registered users may therefore run new experiments on our servers, and if the corresponding model and protocol have been made public, the results will also be visible to all.

In the Cardiac Electrophysiology Web Lab models and protocols are simulated automatically on compute nodes in the back-end. A registered user may for example upload a new model and immediately run it using all available protocols. Every combination of model and protocol represents a single simulation job. This job including required files and annotations are packed into a COMBINE archive, see Section 4.2, and sent to the compute node. The simulation results are then added to that COMBINE archive and reported back to the front-end. Behind the scenes, a database stores model and protocol descriptions along with the cached results of the corresponding experiments (every protocol has been run on every compatible model, i.e., every model that contains the biological quantities being probed by the protocol). Models, protocols, and results may be viewed by anyone, with plots of the results rendered in the web browser. In addition, any experiments may be compared, combining their results in an interactive, web-based graphic. The database also stores earlier versions of models, protocols, and simulation results. Different versions can be compared in the Web Lab, e.g., using the BiVeS tool, compare Sections 2.3, 3.1 and 3.2.3. The underlying simulation environment and the Web Lab portal code have been released as open source²⁵ and can be accessed via the web portal, as can the documentation on using the system, uploading your own models, and writing your own protocols²⁶.

Figure 4.9 displays the experiment overview table. Results are color coded according to the experiment's state, i.e., queued, running, inapplicable (the protocol's required quantities are not present, or not labelled, in the model), failed to run (usually due to numerical instabilities; see below), partially finished (some post-processing was not possible), or successfully finished. Note that I do not compare simulated results against experimental data, and hence the colour coding does not represent model correctness or agreement with experimental data in any sense; it simply indicates the degree to which the simulation experiment was able to be run. Accordingly, a model displaying all green results should not be considered as the best model. Also note that these virtual experiments are not performed on the fly when results are viewed. Some protocols require extensive simulation and post-processing, and thus take a substantial amount of time to run. Instead, experiments are run by the Web Lab when new (or updated) models or protocols are added to the system, and the results are cached.

The following sections will show some results of individual virtual experiments, highlighting the ways in which different models (or different hypotheses) can make very different predictions. This illuminates certain areas that will require careful attention in cardiac electrophysiology modelling.

²⁵bitbucket.org/joncooper/fcweb, accessed 16 July 2017

²⁶travis.cs.ox.ac.uk/FunctionalCuration/about.html, accessed 16 July 2017

Chapter 4. Support for shareable and reproducible simulation studies

	Extracellular potassium variation	Graph state	ICaL block	ICaL IV Curve	IK1 block	IK1 IV Curve	IKr Block	IKr IV Curve	IKs block	IKs IV Curve	INa block	INa IV Curve	INa late	Ito block	Ito fast block	Ito slow block	NCX Block	RyR Block	S1-S2 Restitution	Steady state 0.5Hz pacing	Steady state 1Hz pacing	Steady state 2Hz pacing	Steady state 3Hz pacing	Steady state 4Hz pacing	Steady State Restitution
Aslanidi atrial model 2009																									
Aslanidi Purkinje model 2009																									
Beeler-Reuter 1977																									
Benson 2008																									
Bernus 2002																									
Bondarenko 2004 Apical																									
Carro 2011 Endo																									
Carro 2011 Epi																									
Clancy Rudy 2002																									
Courtemanche 98																									
Decker 2009																									
DiFrancesco Noble 1985																									
Earm Noble 1990																									
Faber Rudy 2000																									
Fink 2008																									
Grandi 2010 Endo																									
Grandi 2010 Epi																									
Iyer 04																									
Iyer 07																									
Li 2010																									
Luo Rudy 1991																									
Luo Rudy 1994																									
Mahajan 2008																									
Maleckar 2009																									
Matsuoka 2003																									
Noble 1998																									
Noble 91																									
O'Hara 2011 Endo																									
O'Hara 2011 Epi																									
Priebe 1998																									
Shannon 2004																									
ten Tusscher 2004 Endo																									
ten Tusscher 2004 Epi																									
ten Tusscher 2006 Endo																									
ten Tusscher 2006 Epi																									
Winslow 99																									

Figure 4.9. Overview of the virtual experiments available at the Web Lab. Each square represents the stored results of a single virtual experiment, colour coded according to status. Green indicates that the protocol ran to completion, orange that it did not complete but some of the expected graphs are nevertheless available (so only a subset of the simulations and/or post-processing failed), red that no graphs are available, and grey that the model and protocol are incompatible (i.e., the model does not contain some biological feature probed by the protocol). Note that the colours do not indicate model correctness in any sense.

4.6.2. Exploring model characteristics and discovering steady states

For the first time cardiac electrophysiology researchers can easily examine the AP waveforms produced by different models. Figure 4.10 shows a snapshot of APs for several human ventricular models at both 1 Hz²⁷ and 2 Hz²⁸. Note that these voltage traces are not the only outputs produced by the corresponding protocol: Other outputs can be viewed and compared online by selecting the appropriate action icons below the plots. For instance, the calcium transients corresponding to the APs in Figure 4.10 can be compared at s.binfalse.de/CA1Hz (1 Hz) and s.binfalse.de/CA2Hz (2 Hz). Furthermore, it is easily possible to evaluate model behaviours under more complex protocols (e.g., S1-S2 or steady-state restitution curves), as shown in Figure 4.11 for the O'Hara 2011 models²⁹ [OHa+11].

Although a large number of models include dynamic changes in ionic concentrations (first introduced by DiFrancesco and Noble, [DN85] in 1985), ionic homoeostasis would appear to be one of the more controversial areas, as evidenced by the wide variety of model responses (or hypothesis predictions) to alterations of this system. For example, Figure 4.12 presents the (steady-state) effect on the AP duration (APD) of progressive block of the sodium-calcium exchanger (NCX), implemented by scaling its maximum current density. The models make a wide range of predictions, reflecting the current limitations of our knowledge regarding intracellular sodium and calcium homoeostasis [BC15]. Therefore, an appropriate model for any study involving changes to NCX conductance should be selected and evaluated carefully. The Web Lab can assist in this by demonstrating how different models behave. In fact, the model behaviours presented here are only intended to give an impression of the powerful approaches that the Web Lab facilitates.

4.6.3. Correcting errors in model encodings

A discussion about the results of the Decker 2009 model S1-S2 restitution curve (as published in the pilot study, [CMN11]) with the senior author, Prof. Y. Rudy, led to a careful comparison of the results with those in the original model publication [Dec+09]. The differences uncovered an error in the CellML implementation of the Decker model, which had been available since March 2010³⁰. The CellML file was corrected and is now providing an accurate representation of the model to the community³¹. Both, the old³² and the new³³ version are available in the Web Lab³⁴. Differences between the original and corrected model versions can be displayed in the Web Lab by using the BiVeS tool. Figure 3.3 from the previous chapter shows just this

²⁷Interactive visualisation available at s.binfalse.de/AP1Hz, accessed 16 July 2017

²⁸Interactive visualisation available at s.binfalse.de/AP2Hz, accessed 16 July 2017

²⁹Interactive visualisation available at s.binfalse.de/rest-curve, accessed 16 July 2017

³⁰For full details see mirams.wordpress.com/2013/10/22/importance-of-curating-models, accessed 16 July 2017

³¹See simulation results in s.binfalse.de/decker-s1s2 or s.binfalse.de/decker-ap, accessed 16 July 2017

³²Incorrect version of the Decker 2009 model s.binfalse.de/decker-buggy, accessed 16 July 2017

³³Fixed version of the Decker 2009 model s.binfalse.de/decker-fixed, accessed 16 July 2017

³⁴All versions of the Decker 2009 model s.binfalse.de/decker-versions, accessed 16 July 2017

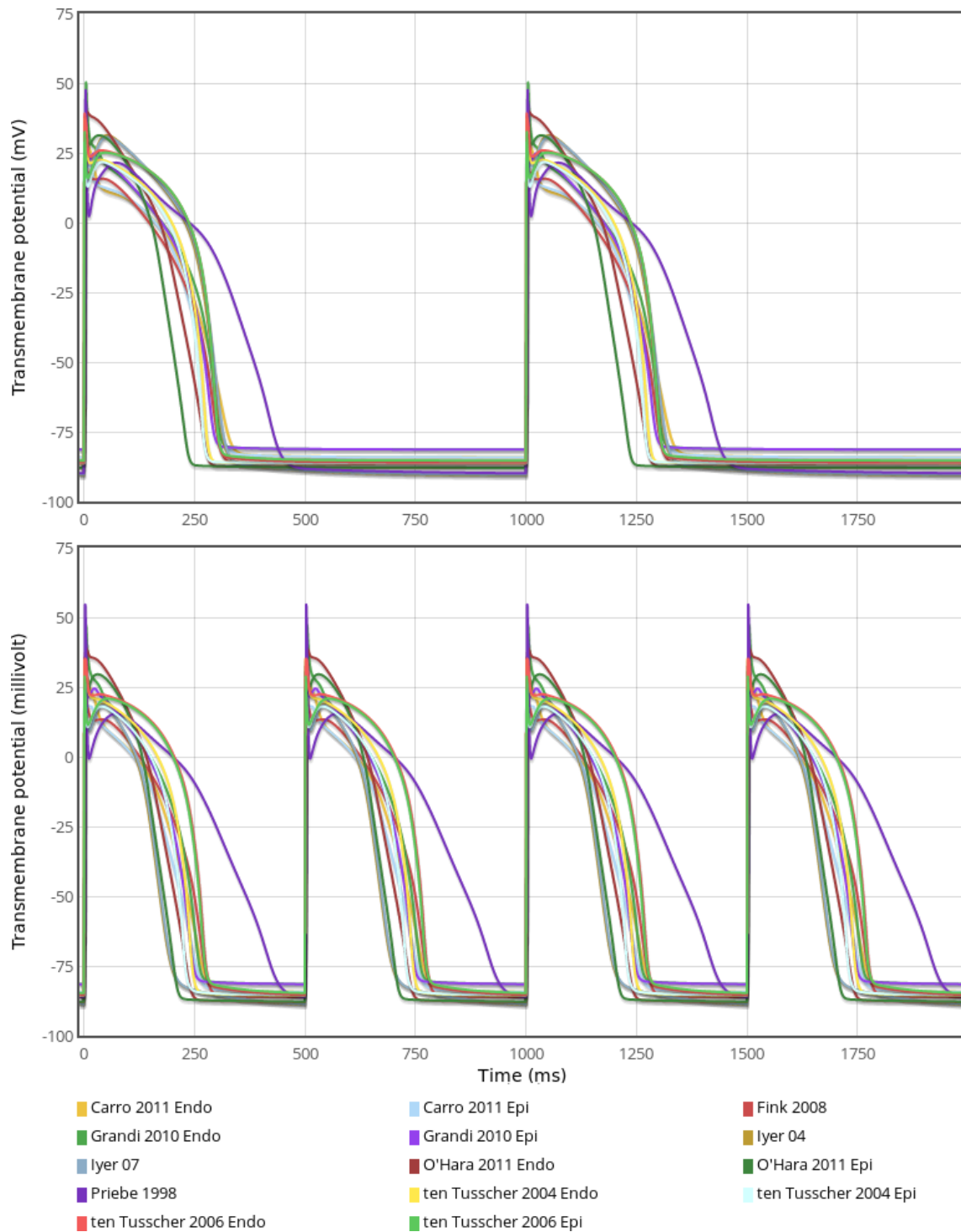


Figure 4.10. Comparison of AP waveforms. 1 Hz (top) and 2 Hz (bottom) steady-pacing AP waveforms for a selection of human ventricular cell models. Both plots show the behaviour of 14 different models under the same conditions following a fixed protocol. Interactive visualisations are available online at s.binfalse.de/AP1Hz (1 Hz) and s.binfalse.de/AP2Hz (2 Hz).

outputs_Restitution_curve_gnuplot_data.csv

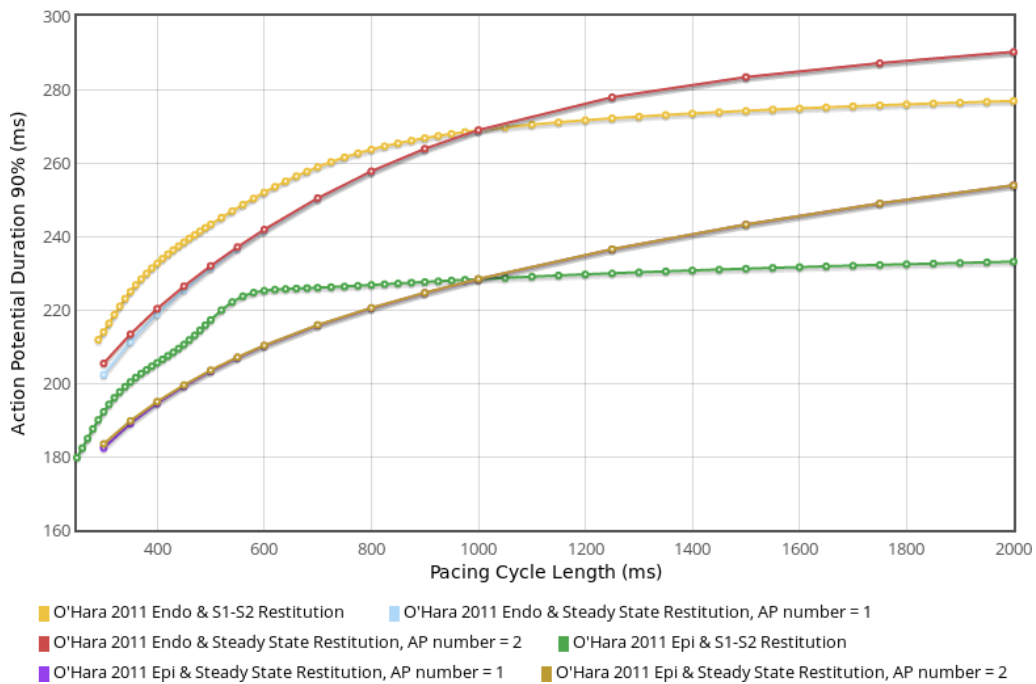


Figure 4.11. Restitution curves for the O'Hara 2011 model epi- and endocardial variants.

Variation in APD at 90% repolarisation is shown for the S1-S2 protocol with the initial stimulus interval S1 set to 1000 ms, and for steady-state restitution (in which two paces are analysed and plotted as two lines, to show fork or alternans at short rates, visible in the endocardial variant). This demonstrates the Web Lab's ability to run complex protocols with intricate post-processing. Interactive visualisations are available online at s.binfalse.de/rest-curve.

comparison³⁵. These differences only become apparent when a model is tested in a range of situations, which is finally possible through the Web Lab.

4.6.4. Impact and discussion

In the previous sections I presented a new online resource for users and developers of mathematical models of cardiac electrophysiology. As shown, it offers great flexibility for analysing and comparing models under different experimental conditions. This will help model users to select suitable models for their simulation studies by ensuring that relevant basic behaviour can be reproduced (for instance, that a model intended for use in simulating arrhythmia has suitable restitution properties). It can also highlight models whose implementations have problems with numerical stability, or those that drift to non-physiological regimes.

The Web Lab will also be of benefit to model developers. They may upload their in-development models to the system, keeping them private if needed, to evaluate their behaviour against a much wider range of protocols than are typically considered during construction of

³⁵Rerun the comparison online s.binfalse.de/decker-diff, accessed 16 July 2017

outputs_Relative_APD90_gnuplot_data.csv

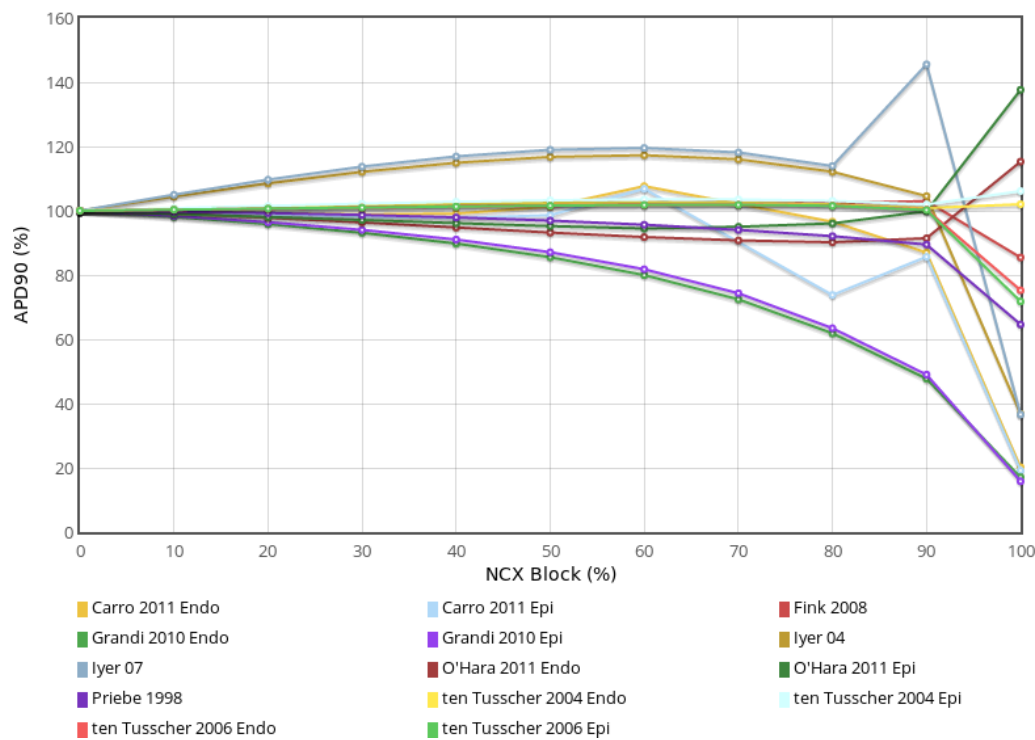


Figure 4.12. Effect of blockade of NCX on steady-state APD in some human ventricular cell models. Note that across 0–80% NCX block, some models predict little effect (<5% change), whereas others predict 20% prolongation and still others predict 20% shortening. At 80–100% block, the results vary dramatically, with models predicting effects ranging from 45% prolongation to 20% shortening compared with control. Interactive visualisations are available online at s.binfal.se/de/NCX-block.

a new model. If a particular experiment is not already available, the corresponding protocol may be submitted as well. New model versions may be uploaded until the desired set of behavioural characteristics is obtained, and the final model can be made public when it is published. The publication could even refer to the stored results as evidence that the model has been thoroughly tested.

Despite the efforts to produce reliable virtual experiments with this system, unexpected behaviour may occur that is not necessarily a real consequence of the model. Mathematical singularities or other numerical simulation issues may cause the simulated experiment to fail, leading to many of the red boxes in Figure 4.9. Sometimes the published representation of the model is in error, or its CellML encoding is (as was the case for the Decker 2009 model discussed above). On other occasions, the protocol, especially the post-processing section, may need further refinement to account for raw simulation results that fall somewhat outside the expected regime – computing a robust APD that accounts for any shape of AP, particularly pathological cases, is surprisingly complex.

Chapter 4. Support for shareable and reproducible simulation studies

As noted above, annotations were used to indicate the physiological meaning of model variables to form the interface between models and protocols, so that a single protocol can be applied to models that may use different names to represent the same concept. Thus, the database contains copies of models from the CellML repository [Llo+08] annotated with terms developed for this purpose. Ideally, these annotations would instead be stored along with the reference versions of the model in the Physiome Model Repository itself, and community-accepted annotations would be used to promote wider interoperability. The direct use of curated models would also address the accountability challenge mentioned above. Related ongoing work is adding more structure to the annotations by defining relationships between terms. This structure could then be used by enhanced tools to provide even more sophisticated interfacing between models and protocols, for instance, by clamping all extracellular concentrations without having to specify which ions may be present in the model.

Other enhancements to the tools, and indeed the protocol language, may also be required as new ideas for protocols arise. Parameter-estimation techniques may be incorporated into this framework, and further automated checking of experiment results could be investigated [PEU14]. It would also be desirable to use a community-accepted standard for protocols rather than a proprietary representation. Features of the new language were therefore proposed for incorporation into future versions of the SED-ML standard being developed by the systems biology community [Wal+11b].

Finally, the most important ingredient that is missing in the current implementation is a direct link to experimental data. Since protocol descriptions should represent experiments that can be performed in a wet lab [Qui+11], it is natural to associate corresponding experimental data sets with each protocol. Simulated experimental results could then be compared automatically against these data sets, to reveal the extent to which different models match the current knowledge of the system. Although public data repositories for electrocardiogram data exist (e.g., PhysioNet [Gol+00]), there is a notable lack of open sources for cell-level electrophysiology data, which could become a serious impediment to progress. Eventually, I envisage that the model descriptions could be associated explicitly with all of the data that were originally used to parametrise them. As new data become available, all relevant models could be validated against them, and even re-parametrised automatically to capture the latest experimental results within a quantitative model [CVW15].

CHAPTER 5

CONCLUSION

Modelling and simulation is a standard approach to investigate complex biological processes. During the past years great achievements in systems biology enhanced our knowledge of biological environments. Discoveries in this field of research are recorded in computational models which encode the structure of biological networks, and describe their temporal and spatial behaviour. Due to tremendous efforts by the research community, the number of openly available models is impressive and still continually increasing. To support the sharing of models and, thus, the reuse of research results, repositories such as BioModels Database and the Physiome Model Repository collect and store models. These repositories provide the infrastructure necessary to maintain model code and associated metadata. Since only accessible models can be reused, such repositories are essential to guarantee transparent research. The distribution of models through these repositories accelerates collaborative research, encourages model reuse, and facilitates transparent research [Wil+16]. Reusing models improves the modelling workflow by reducing errors and saving time. However, model developers and users to date still face a number of questions.

What happened to my model? This is a typical and valid question of a model developer, whose model was just published in a repository after (re)curation, or who received an updated version of his model from a collaborator. Are the changes introduced by the curator or collaborator still in line with the initial assumptions and ideas of the model creator? Moreover, developers improve or extend their models to, for example, test hypotheses. Thus, regular changes in models lead to different versions of a model. Indeed, research results are in constant flux and models are continuously modified. Modellers often maintain lots of directories containing multiple versions of their model files. The model's file names are often postfixed with a date or something like `final-revised_final2`, which is a known problem not limited to

Chapter 5. Conclusion

modelling¹. However, after a surprisingly short amount of time creators of a version often do not remember the reason for its creation anymore. Small changes may be comprehensible. However, as the list of changes increases it becomes harder to understand their relevance.

In 2013 we identified major requirements of a comprehensive version control system for computational models: (i) it must be entailed to the structure of model documents, i.e. XML-aware, (ii) changes must be transparent and versions must be unambiguously identifiable, addressable, and accessible, (iii) changes must be justified [Wal+13]. My algorithm introduced in Chapter 2.2 supports researchers in identifying and understanding changes between versions of models. A sophisticated strategy compares different model documents, and various output formats communicate the differences. BiVeS, the software library that implements the algorithm, is already integrated in the major tools of the systems biology domain, as shown in Chapter 3. Consequently, modellers can now track and compare versions of a model and, thus, they have improved access to the evolution of their models. The COMODI ontology presented in Section 2.4 helps users in grasping the evolution of a model. The study in Section 3.4 presents how models evolve in open repositories. Figure 3.12 demonstrates the evolution of a computational model.

What is the best model for my task? A model user may be searching for a model to study a specific hypothesis, but which model best describes the given data? And which of the model's versions should the user take? As discussed before, the latest version is not necessarily the best version [Mil+11]. BiVeS together with annotations from the COMODI ontology support researchers in examining and filtering the history of a model, as described in Section 2.4. Using the semantic layer to describe changes in a model allows for storing meaning together with possible implications of these changes. Versions, in which only the annotations were changed, can be ignored if the user is primarily interested in the numerical evaluation of differential equations. Similarly, the semantic annotations with terms from the COMODI ontology may help predicting possible effects on the simulation outcome. This already helps to communicate changes and to select relevant versions.

Tracking the evolution of a model plays an important role in supporting the user. Gaining insights into the process of development of a particular model has the potential to increase the confidence in this model and supports the collaboration of distinct research projects dramatically. Consequently, existing model repositories can benefit from extending their software and functionalities with version control. Once processed, the information that is recorded by a version control system enables users to study a model's history and to answer specific questions about the model (compare Figure 1.3). I extended a graph database and demonstrated how models and their versions can be stored along with information on differences, annotations,

¹phdcomics.com/comics/archive/phd101212s.gif, accessed 1 August 2017

and other related information. This concept allows for precise and complex queries with respect to the model's evolution.

The Cardiac Electrophysiology Web Lab takes another step forward, as presented in Section 4.6. The implemented approach enables model benchmarking. A model can be run and analysed using different simulation setups and, vice versa, different models can be benchmarked using a fixed setup. The simulation results can be compared in the web browser. With this approach, models can be evaluated and validated.

How can I reproduce this figure? As discussed in Chapter 1, reproducibility is still a challenge. Not long ago, models and simulation studies were communicated through plain articles in paper format. To (re)run such a study, a researcher needed to reimplement all the differential equations and to copy parameter values, which may have been missing or incorrect in the original publication (e.g. because of typographical errors). As argued previously, the ability to obtain similar results when reproducing an experiment is a central requirement for the advancement of science. The reproducibility of a scientific study depends on the careful description of the original experiment, including the methods and tools used to perform the experiment, the substrate on which to perform the experiment, and the precise experimental setup, including all necessary influences from the environment. With the help of standard formats the situation improved significantly. The systematic description of biological entities and processes helps understanding the model, its scope, and taken assumptions [Cou+11]. Here, formats such as SBML and CellML have several advantages: Models can be simulated, analysed, and visualised using different software tools; models encoded in standard formats may outlive the tool used to create the model; model exchange becomes feasible; and models can more easily be shared, published, and reused. Especially since journals ask the modellers to upload their models to public databases the reproducibility of studies improved a lot. However, due to the increasing complexity of simulation studies plenty of problems persist. Today's studies consist of multiple, heterogeneous, and sometimes distributed data files, making it difficult to exchange complete and thus reproducible results. Managing and sharing multiple files entails a number of difficulties and can be tedious and error-prone. Some files necessary to build or process a model might have moved, or even be deleted, precluding the reproduction of results and ultimately even reusing the model. Approaches such as the COMBINE archive introduced in Section 4.2, can encapsulate everything there is to know about a specific modelling project, including the instructions on how to “open” the archive and interpret it. I developed a Fully Featured COMBINE archive to demonstrate the impact of this approach, see Section 4.3. The tools that I developed and presented in Section 4.4 breath life into COMBINE archives. For example, a web interface provides intuitive access to creating and sharing reproducible simulation studies bundled in COMBINE archives. These simulation studies can then automatically be run and evaluated at the SED-ML Web Tools. Together, my tools support modellers with promoting and

Chapter 5. Conclusion

publishing their work by means of creating, exploring, modifying, and sharing their modelling results in a reproducible and transparent fashion.

In the future, I envision a digital publication, whose graphs have extra buttons to download the corresponding *in silico* experiment, including whatever is necessary to rerun it. The simulation tool of my choice is then able to understand and run the experiment, and eventually reproduces the exact figure from the publication. This would give everyone the power to validate the described findings and to “play” with the virtual experiment. The tools presented in Chapter 4 point in that direction. In fact, the envisioned publication is not too far away! Graphs in a digital publication do not need to be defined in an image format, but can instead be described in a COMBINE archive, which produces the numerical and graphical result. Moreover, the graph may also be interactive using modern web techniques. Two buttons next to the graph may send the user (together with the corresponding COMBINE archive), for example, (i) to the CombineArchiveWeb application, if the user wants to explore and modify the simulation study, or (ii) to the SED-ML Web Tools to immediately rerun the *in silico* experiment and to tinker with parameters. A third button may link to the model’s page in an open repository, where users can get more information and explore the model’s evolution. Finally, the COMBINE archive defining the virtual experiment behind a figure should be properly annotated with provenance information. Specifically, PROV [MG13] and PAV [Cic+13] offer some compelling concepts for model provenance, which modeling tools and platforms should take the responsibility of implementing support for. Equipped with this, readers would know (i) which tool was used to run the experiment, (ii) which researcher(s) created/contributed to the model, (iii) which lab/machine generated the data, and so on. Currently, it is quite an effort and includes plenty of manual work, for just a single figure. However, I am convinced that most of it can be automatised. A reproducible research environment should be able to automatically track the provenance of data, analyses, and results and to package them, for example, as a COMBINE archive.

As proud member of the Free Software Foundation Europe all my software is not just open source, but also freely available. This gives everyone the freedoms² to (i) run, (ii) study, (iii) redistribute, (iv) improve and re-release the tools. Free software helps disseminating tools. Several major repositories already implement and use my tools, emphasising their usefulness and validating the implementations. In addition, BiVeS was also evaluated in a thesis at The University of Manchester, which came to the conclusion

Going forward, the BiVeS differencing software would be the obvious choice [Pot15].

²[fsfe.org/freesoftware/basics/4freedoms.en.html](https://www.fsf.org/freesoftware/basics/4freedoms.en.html), accessed 16 October 2017

CHAPTER 5

BIBLIOGRAPHY

- [Arr11] J Arrowsmith. ‘Trial Watch: Phase II Failures: 2008–2010’. In: *Nature Reviews Drug Discovery* 10.5 (May 2011), pages 328–329. ISSN: [1474-1776](#). DOI: [10.1038/nrd3439](#).
- [Ash+00] M Ashburner, CA Ball, JA Blake, D Botstein et al. ‘Gene Ontology: Tool for the Unification of Biology’. In: *Nature Genetics* 25.1 (May 2000), pages 25–29. DOI: [10.1038/75556](#).
- [BBO10] F Bergmann, F Bergmann and B Olivier. ‘SBML Level 3 Package Proposal: Flux’. In: *Nature Precedings* (24 February 2010). ISSN: [1756-0357](#). DOI: [10.1038/npre.2010.4236.1](#).
- [BC15] DM Bers and Y Chen-Izu. ‘Sodium and Calcium Regulation in Cardiac Myocytes: From Molecules to Heart Failure and Arrhythmia’. In: *The Journal of Physiology* 593.6 (15 March 2015), pages 1327–1329. ISSN: [1469-7793](#). DOI: [10.1113/JP270133](#).
- [BE12] CG Begley and LM Ellis. ‘Drug Development: Raise Standards for Preclinical Cancer Research’. In: *Nature* 483.7391 (29 March 2012), pages 531–533. ISSN: [0028-0836](#). DOI: [10.1038/483531a](#).
- [Bea+09] DA Beard, R Britten, MT Cooling, A Garny et al. ‘CellML Metadata Standards, Associated Tools and Repositories’. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 367.1895 (28 May 2009), pages 1845–1867. ISSN: [1364-503X](#), [1471-2962](#). DOI: [10.1098/rsta.2008.0310](#). PMID: [19380315](#).

Bibliography

- [Bec+10] S Bechhofer, DD Roure, M Gamble, C Goble et al. ‘Research Objects: Towards Exchange and Reuse of Digital Knowledge’. In: *The Future of the Web for Collaborative Science* (2010).
- [Bec+13] S Bechhofer, I Buchan, DD Roure, P Missier et al. ‘Why Linked Data Is Not Enough for Scientists’. In: *Future Generation Computer Systems* 29.2 (2013). Special section: Recent advances in e-Science, pages 599–611. ISSN: [0167-739X](#). DOI: [10.1016/j.future.2011.08.004](#).
- [Bec09] S Bechhofer. ‘OWL: Web Ontology Language’. In: *Encyclopedia of Database Systems*. Edited by L LIU and MT ÖZSU. Springer US, 2009, pages 2008–2009. DOI: [10.1007/978-0-387-39940-9_1073](#).
- [Bel+09] AW Bell, EW Deutsch, CE Au, RE Kearney et al. ‘A HUPO Test Sample Study Reveals Common Problems in Mass Spectrometry-based Proteomics’. In: *Nature Methods* 6.6 (June 2009), pages 423–430. ISSN: [1548-7091](#). DOI: [10.1038/nmeth.1333](#).
- [Ber+14] F Bergmann, R Adams, S Moodie, J Cooper et al. ‘COMBINE Archive and OMEX Format: One File to Share All Information to Reproduce a Modeling Project’. In: *BMC Bioinformatics* 15.1 (2014), page 369. ISSN: [1471-2105](#). DOI: [10.1186/s12859-014-0369-z](#).
- [Ber+15] FT Bergmann, J Cooper, N Le Novère, D Nickerson et al. ‘Simulation Experiment Description Markup Language (SED-ML) Level 1 Version 2’. In: *Journal of Integrative Bioinformatics* 12.2 (4 September 2015), page 262. ISSN: [1613-4516](#). DOI: [10.2390/biecoll-jib-2015-262](#). PMID: [26528560](#).
- [Ber+17] FT Bergmann, D Nickerson, D Waltemath and M Scharm. ‘SED-ML Web Tools: Generate, Modify and Export Standard-Compliant Simulation Studies’. In: *Bioinformatics* (2 January 2017), btw812. ISSN: [1367-4803](#), [1460-2059](#). DOI: [10.1093/bioinformatics/btw812](#).
- [Ber11] F Bergmann. ‘SED-ML Script Language’. In: *Nature Precedings* (2011). DOI: [10.1038/npre.2011.6105.1](#).
- [BI15] CG Begley and JPA Ioannidis. ‘Reproducibility in Science’. In: *Circulation Research* 116.1 (2 January 2015), pages 116–126. ISSN: [0009-7330](#), [1524-4571](#). DOI: [10.1161/CIRCRESAHA.114.303819](#). PMID: [25552691](#).
- [BL97] RC Burns and DDE Long. ‘Efficient Distributed Backup with Delta Compression’. In: *Proceedings of the Fifth Workshop on I/O in Parallel and Distributed Systems*. IOPADS ’97. New York, NY, USA: ACM, 1997, pages 27–36. ISBN: [978-0-89791-966-1](#). DOI: [10.1145/266220.266223](#). URL: <http://doi.acm.org/10.1145/266220.266223>.

- [Bla14] R Blaustein. ‘Reproducibility Undergoes Scrutiny’. In: *BioScience* 64.4 (1 April 2014), pages 368–368. ISSN: 0006-3568, 1525-3244. DOI: [10.1093/biosci/biu017](https://doi.org/10.1093/biosci/biu017).
- [BOH11] M Bostock, V Ogievetsky and J Heer. ‘D3 Data-Driven Documents’. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (December 2011), pages 2301–2309. ISSN: 1077-2626. DOI: [10.1109/TVCG.2011.185](https://doi.org/10.1109/TVCG.2011.185).
- [BR04] JBL Bard and SY Rhee. ‘Ontologies in Biology: Design, Applications and Future Challenges’. In: *Nat Rev Genet* 5.3 (March 2004), pages 213–222. DOI: [10.1038/nrg1295](https://doi.org/10.1038/nrg1295).
- [Bra+01a] U Brandes, M Eiglsperger, I Herman, M Himsolt et al. ‘GraphML Progress Report Structural Layer Proposal’. In: *Graph Drawing*. Springer, Berlin, Heidelberg, 23 September 2001, pages 501–512. DOI: [10.1007/3-540-45848-4_59](https://doi.org/10.1007/3-540-45848-4_59). URL: https://link.springer.com/chapter/10.1007/3-540-45848-4_59.
- [Bra+01b] A Brazma, P Hingamp, J Quackenbush, G Sherlock et al. ‘Minimum Information about a Microarray Experiment (MIAME)—toward Standards for Microarray Data’. In: *Nature Genetics* 29.4 (December 2001), pages 365–371. ISSN: 1061-4036. DOI: [10.1038/ng1201-365](https://doi.org/10.1038/ng1201-365).
- [BS06] FT Bergmann and HM Sauro. ‘SBW - a Modular Framework for Systems Biology’. In: *Proceedings of the 38th Conference on Winter Simulation*. WSC ’06. Monterey, California: Winter Simulation Conference, 2006, pages 1637–1645. ISBN: 978-1-4244-0501-5. URL: <http://dl.acm.org/citation.cfm?id=1218112.1218411>.
- [CAH02] G Cobena, T Abdessalem and Y Hinnach. ‘A Comparative Study for XML Change Detection’. In: *BDA*. 2002. URL: <https://dblp.org/rec/conf/bda/CobenaAH02>.
- [Cal+07] L Calzone, D Thieffry, JJ Tyson and B Novak. ‘Dynamical Modeling of Syncytial Mitotic Cycles in Drosophila Embryos’. In: *Mol. Syst. Biol.* 3 (2007), page 131. DOI: [10.1038/msb4100171](https://doi.org/10.1038/msb4100171).
- [CF07] EM Cherry and FH Fenton. ‘A Tale of Two Dogs: Analyzing Two Models of Canine Ventricular Electrophysiology’. In: *American Journal of Physiology - Heart and Circulatory Physiology* 292.1 (1 January 2007), H43–H55. ISSN: 0363-6135, 1522-1539. DOI: [10.1152/ajpheart.00955.2006](https://doi.org/10.1152/ajpheart.00955.2006). PMID: 16997886.
- [CF10] A Casadevall and FC Fang. ‘Reproducible Science’. In: *Infect. Immun.* 78.12 (December 2010), pages 4972–4975. DOI: [10.1128/IAI.00908-10](https://doi.org/10.1128/IAI.00908-10).
- [Cha+96] SS Chawathe, A Rajaraman, H Garcia-Molina and J Widom. ‘Change Detection in Hierarchically Structured Information’. In: *SIGMOD Rec.* 25.2 (June 1996). ladiff, pages 493–504. ISSN: 0163-5808. DOI: [10.1145/235968.233366](https://doi.org/10.1145/235968.233366).

Bibliography

- [Cic+13] P Ciccarese, S Soiland-Reyes, K Belhajjame, AJ Gray et al. 'PAV Ontology: Provenance, Authoring and Versioning'. In: *J Biomed Sem* 4.1 (2013), page 37. DOI: [10.1186/2041-1480-4-37](https://doi.org/10.1186/2041-1480-4-37).
- [CKS10] T Czauderna, C Klukas and F Schreiber. 'Editing, Validating and Translating of SBGN Maps'. In: *Bioinformatics* 26.18 (15 September 2010), pages 2340–2341. ISSN: [1367-4803](https://doi.org/10.1093/bioinformatics/btq407). DOI: [10.1093/bioinformatics/btq407](https://doi.org/10.1093/bioinformatics/btq407).
- [Cla+11] RH Clayton, O Bernus, EM Cherry, H Dierckx et al. 'Models of Cardiac Tissue Electrophysiology: Progress, Challenges and Open Questions'. In: *Progress in Biophysics and Molecular Biology* 104 (1-3 January 2011), pages 22–48. ISSN: [1873-1732](https://doi.org/10.1016/j.pbiomolbio.2010.05.008). DOI: [10.1016/j.pbiomolbio.2010.05.008](https://doi.org/10.1016/j.pbiomolbio.2010.05.008). PMID: [20553746](https://pubmed.ncbi.nlm.nih.gov/20553746/).
- [CLN13] V Chelliah, C Laibe and NL Novère. 'BioModels Database: A Repository of Mathematical Models of Biological Processes.' In: *Methods Mol Biol* 1021 (2013), pages 189–199. DOI: [10.1007/978-1-62703-450-0_10](https://doi.org/10.1007/978-1-62703-450-0_10). PMID: [23715986](https://pubmed.ncbi.nlm.nih.gov/23715986/).
- [CMN11] J Cooper, GR Mirams and SA Niederer. 'High-Throughput Functional Curation of Cellular Electrophysiology Models'. In: *Progress in Biophysics and Molecular Biology* 107.1 (October 2011), pages 11–20. ISSN: [1873-1732](https://doi.org/10.1016/j.pbiomolbio.2011.06.003). DOI: [10.1016/j.pbiomolbio.2011.06.003](https://doi.org/10.1016/j.pbiomolbio.2011.06.003). PMID: [21704062](https://pubmed.ncbi.nlm.nih.gov/21704062/).
- [CNH05] AA Cuellar, M Nelson and W Hedley. 'CellML Metadata 1.0 Specification — CellML'. 2005. URL: https://www.cellml.org/specifications/metadata/cellml_metadata_1.0 (accessed 14 March 2017).
- [CO13] J Cooper and J Osborne. 'Connecting Models to Data in Multiscale Multicellular Tissue Simulations'. In: *Procedia Computer Science* 18 (2013), pages 712–721. DOI: [10.1016/j.procs.2013.05.235](https://doi.org/10.1016/j.procs.2013.05.235).
- [Coo+11] J Cooper, A Corrias, D Gavaghan and D Noble. 'Considerations for the Use of Cellular Electrophysiology Models within Cardiac Tissue Simulations'. In: *Progress in Biophysics and Molecular Biology* 107.1 (October 2011), pages 74–80. ISSN: [1873-1732](https://doi.org/10.1016/j.pbiomolbio.2011.06.002). DOI: [10.1016/j.pbiomolbio.2011.06.002](https://doi.org/10.1016/j.pbiomolbio.2011.06.002). PMID: [21703295](https://pubmed.ncbi.nlm.nih.gov/21703295/).
- [Cor+12] Ó Corcho, DG Verdejo, K Belhajjame, J Zhao et al. 'Workflow-Centric Research Objects: First Class Citizens in Scholarly Discourse.' In: *Proceedings of Workshop on the Semantic Publishing (SePublica 2012)*. 9th Extended Semantic Web Conference Hersonissos. Crete, Greece, 2012, pages 1–12. URL: http://oa.upm.es/20401/1/INVE_MEM_2012_134375.pdf.
- [Cou+11] M Courtot, N Juty, C Knüpfer, D Waltemath et al. 'Controlled Vocabularies and Semantics in Systems Biology'. In: *Molecular Systems Biology* 7.1 (1 January 2011), page 543. ISSN: [1744-4292](https://doi.org/10.1038/msb.2011.77), [1744-4292](https://doi.org/10.1038/msb.2011.77). DOI: [10.1038/msb.2011.77](https://doi.org/10.1038/msb.2011.77). PMID: [22027554](https://pubmed.ncbi.nlm.nih.gov/22027554/).

- [CSM16] J Cooper, M Scharm and GR Mirams. ‘The Cardiac Electrophysiology Web Lab’. In: *Biophysical Journal* 110.2 (19 January 2016), pages 292–300. ISSN: 0006-3495. DOI: [10.1016/j.bpj.2015.12.012](https://doi.org/10.1016/j.bpj.2015.12.012). PMID: 26789753.
- [Cue+02] A Cuellar, P Nielsen, M Halstead, D Bullivant et al. ‘CellML 1.1 Specification — CellML’. CellML Community, 6 November 2002. URL: https://www.cellml.org/specifications/cellml_1.1 (accessed 22 November 2016).
- [Cue+03] AA Cuellar, CM Lloyd, PF Nielsen, DP Bullivant et al. ‘An Overview of CellML 1.1, a Biological Model Description Language’. In: *SIMULATION* 79.12 (2003), pages 740–747. DOI: [10.1177/0037549703040939](https://doi.org/10.1177/0037549703040939).
- [CVW15] J Cooper, JO Vik and D Waltemath. ‘A Call for Virtual Experiments: Accelerating the Scientific Process’. In: *Progress in Biophysics and Molecular Biology* 117.1 (January 2015), pages 99–106. ISSN: 1873-1732. DOI: [10.1016/j.pbiomolbio.2014.10.001](https://doi.org/10.1016/j.pbiomolbio.2014.10.001). PMID: 25433232.
- [Dad+10] JO Dada, I Spasić, NW Paton and P Mendes. ‘SBRML: A Markup Language for Associating Systems Biology Data with Models’. In: *Bioinformatics* 26.7 (1 April 2010), pages 932–938. ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btq069](https://doi.org/10.1093/bioinformatics/btq069).
- [DCM12] DCMI Usage Board. ‘DCMI Metadata Terms’. Dublin Core Metadata Initiative, 14 June 2012. URL: <http://dublincore.org/documents/dcmi-terms/> (accessed 3 April 2017).
- [Dec+09] KF Decker, J Heijman, JR Silva, TJ Hund et al. ‘Properties and Ionic Mechanisms of Action Potential Adaptation, Restitution, and Accommodation in Canine Epicardium’. In: *American Journal of Physiology - Heart and Circulatory Physiology* 296.4 (1 April 2009), H1017–H1026. ISSN: 0363-6135, 1522-1539. DOI: [10.1152/ajpheart.01216.2008](https://doi.org/10.1152/ajpheart.01216.2008). PMID: 19168720.
- [Def+14] A Defauw, N Vandersickel, P Dawyndt and AV Panfilov. ‘Small Size Ionic Heterogeneities in the Human Heart Can Attract Rotors’. In: *American Journal of Physiology - Heart and Circulatory Physiology* 307.10 (15 November 2014), H1456–H1468. ISSN: 0363-6135, 1522-1539. DOI: [10.1152/ajpheart.00410.2014](https://doi.org/10.1152/ajpheart.00410.2014). PMID: 25217650.
- [Deg+07] K Degtyarenko, P de Matos, M Ennis, J Hastings et al. ‘ChEBI: A Database and Ontology for Chemical Entities of Biological Interest’. In: *Nucleic Acids Research* 36 (Database 23 December 2007), pages D344–D350. ISSN: 0305-1048, 1362-4962. DOI: [10.1093/nar/gkm791](https://doi.org/10.1093/nar/gkm791).
- [DG10] D De Roure and C Goble. ‘Anchors in Shifting Sand: The Primacy of Method in the Web of Data’. In: *WebSci10: Extending the Frontiers of Society On-Line*. Raleigh, NC: US, 2010. URL: <http://journal.webscience.org/325/>.

Bibliography

- [DN85] D DiFrancesco and D Noble. ‘A Model of Cardiac Electrical Activity Incorporating Ionic Pumps and Concentration Changes’. In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 307.1133 (10 January 1985), pages 353–398. ISSN: 0962-8436, 1471-2970. DOI: [10.1098/rstb.1985.0001](https://doi.org/10.1098/rstb.1985.0001). PMID: [2578676](https://pubmed.ncbi.nlm.nih.gov/2578676/).
- [DP14] A Dräger and BØ Palsson. ‘Improving Collaboration by Standardization Efforts in Systems Biology’. In: *Frontiers in Bioengineering and Biotechnology* 2 (2014), page 61. DOI: [10.3389/fbioe.2014.00061](https://doi.org/10.3389/fbioe.2014.00061).
- [Drä+09] A Dräger, H Planatscher, D Motsou Wouamba, A Schröder et al. ‘SBML2LaTEX: Conversion of SBML Files into Human-Readable Reports’. In: *Bioinformatics* 25.11 (1 June 2009), pages 1455–1456. ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btp170](https://doi.org/10.1093/bioinformatics/btp170).
- [EL00] MB Elowitz and S Leibler. ‘A Synthetic Oscillatory Network of Transcriptional Regulators’. In: *Nature* 403.6767 (20 January 2000), pages 335–338. ISSN: 0028-0836. DOI: [10.1038/35002125](https://doi.org/10.1038/35002125).
- [Ell+01] J Ellson, E Gansner, L Koutsofios, SC North et al. ‘Graphviz— Open Source Graph Drawing Tools’. In: *Graph Drawing*. Springer, Berlin, Heidelberg, 23 September 2001, pages 483–484. DOI: [10.1007/3-540-45848-4_57](https://doi.org/10.1007/3-540-45848-4_57). URL: https://link.springer.com/chapter/10.1007/3-540-45848-4_57.
- [Err+14] TM Errington, E Iorns, W Gunn, FE Tan et al. ‘An Open Investigation of the Reproducibility of Cancer Biology Research’. In: *eLife* 3 (10 December 2014), e04333. ISSN: 2050-084X. DOI: [10.7554/eLife.04333](https://doi.org/10.7554/eLife.04333). PMID: [25490932](https://pubmed.ncbi.nlm.nih.gov/25490932/).
- [FCI17] D Fanelli, R Costas and JPA Ioannidis. ‘Meta-Assessment of Bias in Science’. In: *Proceedings of the National Academy of Sciences* (20 March 2017), page 201618569. ISSN: 0027-8424, 1091-6490. DOI: [10.1073/pnas.1618569114](https://doi.org/10.1073/pnas.1618569114). PMID: [28320937](https://pubmed.ncbi.nlm.nih.gov/28320937/).
- [Fed12] S Federhen. ‘The NCBI Taxonomy Database’. In: *Nucleic Acids Research* 40 (D1 1 January 2012), pages D136–D143. ISSN: 0305-1048, 1362-4962. DOI: [10.1093/nar/gkr1178](https://doi.org/10.1093/nar/gkr1178).
- [Fie+08] D Field, G Garrity, T Gray, N Morrison et al. ‘The Minimum Information about a Genome Sequence (MIGS) Specification’. In: *Nature Biotechnology* 26.5 (May 2008), pages 541–547. ISSN: 1087-0156. DOI: [10.1038/nbt1360](https://doi.org/10.1038/nbt1360).
- [Fin+04] A Finkelstein, J Hetherington, Linzhong Li, O Margoninski et al. ‘Computational Challenges of Systems Biology’. In: *Computer* 37.5 (May 2004), pages 26–33. ISSN: 0018-9162. DOI: [10.1109/MC.2004.1297236](https://doi.org/10.1109/MC.2004.1297236).

- [FK05] N Freed and JC Klensin. ‘Media Type Specifications and Registration Procedures’. The Internet Society, 2005. URL: <https://tools.ietf.org/html/rfc4288> (accessed 3 April 2017).
- [Fun+03] A Funahashi, M Morohashi, H Kitano and N Tanimura. ‘CellDesigner: A Process Diagram Editor for Gene-Regulatory and Biochemical Networks’. In: *BIOSILICO* 1.5 (November 2003), pages 159–162. DOI: [10.1016/s1478-5382\(03\)02370-9](https://doi.org/10.1016/s1478-5382(03)02370-9).
- [Gar+08] A Garny, DP Nickerson, J Cooper, RW dos Santos et al. ‘CellML and Associated Tools and Techniques’. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 366.1878 (13 September 2008), pages 3017–3043. ISSN: [1364-503X](https://doi.org/10.1098/rsta.2008.0094), [1471-2962](https://doi.org/10.1098/rsta.2008.0094). DOI: [10.1098/rsta.2008.0094](https://doi.org/10.1098/rsta.2008.0094). PMID: [18579471](https://pubmed.ncbi.nlm.nih.gov/18579471/).
- [Gar+13] D Garijo, S Kinnings, L Xie, L Xie et al. ‘Quantifying Reproducibility in Computational Biology: The Case of the Tuberculosis Drugome’. In: *PLOS ONE* 8.11 (27 November 2013), e80278. ISSN: [1932-6203](https://doi.org/10.1371/journal.pone.0080278). DOI: [10.1371/journal.pone.0080278](https://doi.org/10.1371/journal.pone.0080278).
- [Gau+13] R Gauges, U Rost, S Sahle, K Wengler et al. ‘SBML Level 3 Layout Package Version 1 Release 1’. SBML Community, 13 August 2013. URL: <http://co.mbine.org/specifications/sbml.level-3.version-1.layout.version-1.release-1> (accessed 20 April 2017).
- [Gau+15] R Gauges, U Rost, S Sahle, K Wengler et al. ‘The Systems Biology Markup Language (SBML) Level 3 Package: Layout, Version 1 Core’. In: *Journal of Integrative Bioinformatics* 12.2 (4 September 2015), page 267. ISSN: [1613-4516](https://doi.org/10.2390/biecoll-jib-2015-267). DOI: [10.2390/biecoll-jib-2015-267](https://doi.org/10.2390/biecoll-jib-2015-267). PMID: [26528565](https://pubmed.ncbi.nlm.nih.gov/26528565/).
- [Gen+11] JH Gennari, ML Neal, M Galdzicki and DL Cook. ‘Multiple Ontologies in Action: Composite Annotations for Biosimulation Models’. In: *Journal of Biomedical Informatics* 44.1 (February 2011), pages 146–154. DOI: [10.1016/j.jbi.2010.06.007](https://doi.org/10.1016/j.jbi.2010.06.007).
- [Gen05] R Gentleman. ‘Reproducible Research: A Bioinformatics Case Study’. In: *Stat Appl Genet Mol Biol* 4.1 (2005), Article2. DOI: [10.2202/1544-6115.1034](https://doi.org/10.2202/1544-6115.1034).
- [GH13] A Gupta and TA Henzinger, editors. ‘Computational Methods in Systems Biology’. Redacted by D Hutchison, T Kanade, J Kittler, JM Kleinberg et al. Volume 8130. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: [978-3-642-40707-9](https://doi.org/10.1007/978-3-642-40707-9). URL: <http://link.springer.com/10.1007/978-3-642-40707-9>.

Bibliography

- [Gle+10] P Gleeson, S Crook, RC Cannon, ML Hines et al. ‘NeuroML: A Language for Describing Data Driven Models of Neurons and Networks with a High Degree of Biological Detail’. In: *PLoS Comput Biol* 6.6 (June 2010), e1000815. DOI: [10.1371/journal.pcbi.1000815](https://doi.org/10.1371/journal.pcbi.1000815).
- [Gob02] C Goble. ‘Position Statement: Musings on Provenance, Workflow and (Semantic Web) Annotations for Bioinformatics’. In: *Workshop on Data Derivation and Provenance*. Chicago, Illinois, 2002.
- [Gol+00] AL Goldberger, LA Amaral, L Glass, JM Hausdorff et al. ‘PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals’. In: *Circulation* 101.23 (13 June 2000), E215–220. ISSN: [1524-4539](https://doi.org/10.1161/01.CIR.101.23.e215). DOI: [10.1161/01.CIR.101.23.e215](https://doi.org/10.1161/01.CIR.101.23.e215). PMID: [10851218](https://pubmed.ncbi.nlm.nih.gov/10851218/).
- [Gri+06] V Grimm, U Berger, F Bastiansen, S Eliassen et al. ‘A Standard Protocol for Describing Individual-Based and Agent-Based Models’. In: *Ecological Modelling* 198 (1–2 15 September 2006), pages 115–126. ISSN: [0304-3800](https://doi.org/10.1016/j.ecolmodel.2006.04.023). DOI: [10.1016/j.ecolmodel.2006.04.023](https://doi.org/10.1016/j.ecolmodel.2006.04.023).
- [Hel+02] DM Hellerstein, YY Goland, A Feldmann, JC Mogul et al. ‘Delta Encoding in HTTP’. The Internet Society, 2002. URL: <https://tools.ietf.org/html/rfc3229> (accessed 4 May 2017).
- [Hen+10] R Henkel, L Endler, A Peters, NL Noverre et al. ‘Ranked Retrieval of Computational Biology Models’. In: *BMC Bioinformatics* 11.1 (2010), page 423. ISSN: [1471-2105](https://doi.org/10.1186/1471-2105-11-423). DOI: [10.1186/1471-2105-11-423](https://doi.org/10.1186/1471-2105-11-423).
- [Hin+04] ML Hines, T Morse, M Migliore, NT Carnevale et al. ‘ModelDB: A Database to Support Computational Neuroscience’. In: *Journal of Computational Neuroscience* 17.1 (2004), pages 7–11. ISSN: [0929-5313, 1573-6873](https://doi.org/10.1023/B:JCNS.0000023869.22017.2e). DOI: [10.1023/B:JCNS.0000023869.22017.2e](https://doi.org/10.1023/B:JCNS.0000023869.22017.2e).
- [HM76] JW Hunt and MD McIlroy. ‘An Algorithm for Differential File Comparison’. In: (CSTR 41 1976). URL: <http://www1.cs.dartmouth.edu/%E2%88%BCdoug/diff.ps>.
- [HN01] W Hedley and M Nelson. ‘CellML 1.0 Specification — CellML’. CellML Community, 10 August 2001. URL: https://www.cellml.org/specifications/cellml_1.0 (accessed 22 November 2016).
- [Hol+12] HG Holzhütter, D Drasdo, T Preusser, J Lippert et al. ‘The Virtual Liver: A Multidisciplinary, Multilevel Challenge for Systems Biology’. In: *Wiley Interdisciplinary Reviews: Systems Biology and Medicine* 4.3 (1 May 2012), pages 221–235. ISSN: [1939-005X](https://doi.org/10.1002/wsbm.1158). DOI: [10.1002/wsbm.1158](https://doi.org/10.1002/wsbm.1158).

- [Hoo+06] S Hoops, S Sahle, R Gauges, C Lee et al. ‘COPASI – a COMplex PATHway SIMulator’. In: *Bioinformatics* 22.24 (2006), pages 3067–3074. DOI: [10.1093/bioinformatics/btl485](https://doi.org/10.1093/bioinformatics/btl485).
- [HP02] FC Hoppensteadt and CS Peskin. ‘Modeling and Simulation in Medicine and the Life Sciences’. Volume 10. Texts in Applied Mathematics. New York, NY: Springer New York, 2002. ISBN: [978-1-4419-2871-9](https://doi.org/10.1007/978-0-387-21571-6). URL: <http://link.springer.com/10.1007/978-0-387-21571-6>.
- [HS16a] M Hucka and LP Smith. ‘SBML Level 3 Groups Package Version 1 Release 1’. SBML Community, 1 April 2016. URL: <http://co.mbine.org/standards/sbml/level-3/version-1/groups/version-1/release-1> (accessed 20 April 2017).
- [HS16b] M Hucka and LP Smith. ‘SBML Level 3 Package: Groups, Version 1 Release 1’. In: *Journal of Integrative Bioinformatics* (2016). DOI: [10.2390/biecoll-jib-2016-290](https://doi.org/10.2390/biecoll-jib-2016-290).
- [Huc+03] M Hucka, A Finney, HM Sauro, H Bolouri et al. ‘The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models’. In: *Bioinformatics* 19.4 (2003), pages 524–531. DOI: [10.1093/bioinformatics/btg015](https://doi.org/10.1093/bioinformatics/btg015).
- [Huc+10] M Hucka, M Hucka, F Bergmann, S Hoops et al. ‘The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core’. In: *Nature Precedings* (6 October 2010). ISSN: [1756-0357](https://doi.org/10.1038/npre.2010.4959.1). DOI: [10.1038/npre.2010.4959.1](https://doi.org/10.1038/npre.2010.4959.1).
- [Huc+11] M Hucka, FT Bergmann, SM Keating and LP Smith. ‘A Profile of Today’s SBML-Compatible Software’. In: *Proceedings of the 2011 IEEE Seventh International Conference on E-Science Workshops*. ESCIENCEW ’11. Washington, DC, USA: IEEE Computer Society, 2011, pages 143–150. ISBN: [978-0-7695-4598-1](https://doi.org/10.1109/eScienceW.2011.28). DOI: [10.1109/eScienceW.2011.28](https://doi.org/10.1109/eScienceW.2011.28).
- [Huc+15a] M Hucka, FT Bergmann, A Draeger, S Hoops et al. ‘Systems Biology Markup Language (SBML) Level 2 Version 5: Structures and Facilities for Model Definitions’. In: *Journal of Integrative Bioinformatics* (2015). DOI: [10.2390/biecoll-jib-2015-271](https://doi.org/10.2390/biecoll-jib-2015-271).
- [Huc+15b] M Hucka, DP Nickerson, GD Bader, FT Bergmann et al. ‘Promoting Coordinated Development of Community-Based Information Standards for Modeling in Biology: The COMBINE Initiative’. In: *Frontiers in Bioengineering and Biotechnology* 3 (24 February 2015). ISSN: [2296-4185](https://doi.org/10.3389/fbioe.2015.00019). DOI: [10.3389/fbioe.2015.00019](https://doi.org/10.3389/fbioe.2015.00019).
- [Huc13] M Hucka. ‘A New Language for a New Biology: How SBML and Other Tools Are Transforming Models of Life’. Published: Victorian Systems Biology Symposium, Australia Victorian Systems Biology Symposium, Australia. August 2013.

Bibliography

- [Hun06] PJ Hunter. ‘Modeling Human Physiology: The IUPS/EMBS Physiome Project’. In: *Proceedings of the IEEE* 94.4 (April 2006), pages 678–691. ISSN: 0018-9219. DOI: [10.1109/JPROC.2006.871767](https://doi.org/10.1109/JPROC.2006.871767).
- [HWW15] R Henkel, O Wolkenhauer and D Waltemath. ‘Combining Computational Models, Semantic Annotations and Simulation Experiments in a Graph Database’. In: *Database* 2015 (1 January 2015). DOI: [10.1093/database/bau130](https://doi.org/10.1093/database/bau130).
- [Ier+12] V Iersel, M P, AC Villéger, T Czauderna et al. ‘Software Support for SBGN Maps: SBGN-ML and LibSBGN’. In: *Bioinformatics* 28.15 (1 August 2012), pages 2016–2021. ISSN: 1367-4803. DOI: [10.1093/bioinformatics/bts270](https://doi.org/10.1093/bioinformatics/bts270).
- [IM14] R Iannella and J McKinney. ‘vCard Ontology - for Describing People and Organizations’. w3c, 2014. URL: <https://www.w3.org/TR/vcard-rdf/> (accessed 3 April 2017).
- [Ioa+09] JPA Ioannidis, DB Allison, CA Ball, I Coulibaly et al. ‘Repeatability of Published Microarray Gene Expression Analyses’. In: *Nature Genetics* 41.2 (February 2009), pages 149–155. ISSN: 1061-4036. DOI: [10.1038/ng.295](https://doi.org/10.1038/ng.295).
- [JLL12] N Juty, N Le Novère and C Laibe. ‘Identifiers.Org and MIRIAM Registry: Community Resources to Provide Persistent Identification’. In: *Nucleic Acids Research* 40 (D1 1 January 2012), pages D580–D586. ISSN: 0305-1048, 1362-4962. DOI: [10.1093/nar/gkr1097](https://doi.org/10.1093/nar/gkr1097).
- [Jun+12] A Junker, H Rohn, T Czauderna, C Klukas et al. ‘Creating Interactive, Web-Based and Data-Enriched Maps with the Systems Biology Graphical Notation’. In: *Nat Protoc* 7.3 (March 2012), pages 579–593. DOI: [10.1038/nprot.2012.002](https://doi.org/10.1038/nprot.2012.002).
- [Kar+12] JR Karr, JC Sanghvi, DN Macklin, MV Gutschow et al. ‘A Whole-Cell Computational Model Predicts Phenotype from Genotype’. In: *Cell* 150.2 (20 July 2012), pages 389–401. ISSN: 0092-8674, 1097-4172. DOI: [10.1016/j.cell.2012.05.044](https://doi.org/10.1016/j.cell.2012.05.044). PMID: 22817898.
- [Ket+10] C Kettner, D Field, SA Sansone, C Taylor et al. ‘Meeting Report from the Second “Minimum Information for Biological and Biomedical Investigations” (MIBBI) Workshop’. Report 3. BioMed Central, 31 December 2010, page 259. URL: <http://standardsingenomics.biomedcentral.com/articles/10.4056/sigs.147362> (accessed 21 April 2017).
- [Kin+11] RD King, M Liakata, C Lu, SG Oliver et al. ‘On the Formalization and Reuse of Scientific Research’. In: *J R Soc Interface* 8.63 (October 2011), pages 1440–1448. DOI: [10.1098/rsif.2011.0029](https://doi.org/10.1098/rsif.2011.0029).

- [Kit+05] H Kitano, A Funahashi, Y Matsuoka and K Oda. ‘Using Process Diagrams for the Graphical Representation of Biological Networks’. In: *Nature Biotechnology* 23.8 (August 2005), pages 961–966. ISSN: 1087-0156. DOI: 10.1038/nbt1111.
- [Kli+07] E Klipp, W Liebermeister, A Helbig, A Kowald et al. ‘Systems Biology Standards—the Community Speaks’. In: *Nature Biotechnology* 25.4 (April 2007), pages 390–391. ISSN: 1087-0156. DOI: 10.1038/nbt0407-390.
- [Kli+16] E Klipp, W Liebermeister, C Wierling and A Kowald. ‘Systems Biology: A Textbook’. Wiley-Blackwell, 2016. ISBN: 978-3-527-33636-4. URL: <http://eu.wiley.com/WileyCDA/WileyTitle/productCd-3527336362.html>.
- [Kli16] P Klink. ‘FieldedText’. 2016. URL: <http://www.fieldedtext.org/> (accessed 1 May 2017).
- [Kol+11] FA Kolpakov, NI Tolstykh, TF Valeev, IN Kiselev et al. ‘BioUML – Open Source Plug-in Based Platform for Bioinformatics: Invitation to Collaboration’. In: *Proceedings of the International Moscow Conference on Computational Molecular Biology*. International moscow conference on computational molecular biology. Moscow State University, 2011, 172ff.
- [Kol02] FA Kolpakov. ‘BioUML – Framework for Visual Modeling and Simulation Biological Systems’. In: *Proceedings of the International Conference on Bioinformatics of Genome Regulation and Structure*. International Conference on Bioinformatics of Genome Regulation and Structure. 2002.
- [LHN04] CM Lloyd, MDB Halstead and PF Nielsen. ‘CellML: Its Future, Present and Past’. In: *Progress in Biophysics and Molecular Biology* 85 (2-3 2004), pages 433–450. ISSN: 0079-6107. DOI: 10.1016/j.pbiomolbio.2004.01.004. PMID: 15142756.
- [Li+10] C Li, M Donizelli, N Rodriguez, H Dharuri et al. ‘BioModels Database: An Enhanced, Curated and Annotated Resource for Published Quantitative Kinetic Models’. In: *BMC Systems Biology* 4.1 (2010), page 92. ISSN: 1752-0509. DOI: 10.1186/1752-0509-4-92.
- [Llo+08] CM Lloyd, JR Lawson, PJ Hunter and PF Nielsen. ‘The CellML Model Repository’. In: *Bioinformatics* 24.18 (2008), pages 2122–2123. DOI: 10.1093/bioinformatics/btn390.
- [Lop+10] CT Lopes, M Franz, F Kazi, SL Donaldson et al. ‘Cytoscape Web: An Interactive Web-Based Network Browser’. In: *Bioinformatics* 26 (2010), pages 2347–2348. DOI: 10.1093/bioinformatics/btq430.
- [LS99] O Lassila and RR Swick. ‘Resource Description Framework (RDF) Model and Syntax Specification’. 1999. URL: <https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/> (accessed 14 March 2017).

Bibliography

- [Mar+01] A Marian, S Abiteboul, G Cobena and L Mignet. ‘Change-Centric Management of Versions in an XML Warehouse’. In: *Proceedings of the 27th International Conference on Very Large Data Bases*. VLDB ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pages 581–590. ISBN: 1-55860-804-4. URL: <http://dl.acm.org/citation.cfm?id=645927.672205>.
- [Mar+13] A Mardinoglu, R Agren, C Kampf, A Asplund et al. ‘Integration of Clinical Data with a Genome-Scale Metabolic Model of the Human Adipocyte’. In: *Molecular Systems Biology* 9.1 (1 January 2013), n/a–n/a. ISSN: 1744-4292. DOI: 10.1038/msb.2013.5.
- [MBF05] L Masinter, T Berners-Lee and RT Fielding. ‘Uniform Resource Identifier (URI): Generic Syntax’. The Internet Society, 2005. URL: <https://tools.ietf.org/html/rfc3986> (accessed 3 April 2017).
- [McD+15] RA McDougal, TM Morse, ML Hines and GM Shepherd. ‘ModelView for ModelDB: Online Presentation of Model Structure’. In: *Neuroinformatics* 13.4 (21 April 2015), pages 459–470. ISSN: 1539-2791, 1559-0089. DOI: 10.1007/s12021-015-9269-2.
- [Mes10] JP Mesirov. ‘Computer Science. Accessible Reproducible Research’. In: *Science* 327.5964 (January 2010), pages 415–416. DOI: 10.1126/science.1179653.
- [MG13] L Moreau and P Groth. ‘Provenance: An Introduction to PROV (Synthesis Lectures on the Semantic Web: Theory and Technology)’. Edited by J Hendler and Y Ding. Morgan & Claypool Publishers, September 2013. ISBN: 978-1-62705-221-4. URL: <http://www.provbook.org/>.
- [Mi+15] H Mi, F Schreiber, S Moodie, T Czauderna et al. ‘Systems Biology Graphical Notation: Activity Flow Language Level 1 Version 1.2’. In: *Journal of Integrative Bioinformatics* (2015). DOI: 10.2390/biecoll-jib-2015-265.
- [Mil+11] A Miller, T Yu, R Britten, M Cooling et al. ‘Revision History Aware Repositories of Computational Models of Biological Systems’. In: *BMC Bioinformatics* 12.1 (2011), page 22. DOI: 10.1186/1471-2105-12-22.
- [Mil13] JJ Miller. ‘Graph Database Applications and Concepts with Neo4j’. In: *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*. Volume 2324. 2013, page 36. URL: <https://pdfs.semanticscholar.org/322a/6e1f464330751dea2eb6beecac24466322ad.pdf>.
- [Mir+11] GR Mirams, Y Cui, A Sher, M Fink et al. ‘Simulation of Multiple Ion Channel Block Provides Improved Early Prediction of Compounds’ Clinical Torsadogenic Risk’. In: *Cardiovascular Research* 91.1 (1 July 2011), pages 53–61. ISSN: 0008-6363. DOI: 10.1093/cvr/cvr044. PMID: 21300721.

- [Mir+12] GR Mirams, MR Davies, Y Cui, P Kohl et al. ‘Application of Cardiac Electrophysiology Simulations to Pro-Arrhythmic Safety Testing’. In: *British Journal of Pharmacology* 167.5 (November 2012), pages 932–945. ISSN: 0007-1188. DOI: [10.1111/j.1476-5381.2012.02020.x](https://doi.org/10.1111/j.1476-5381.2012.02020.x). PMID: 22568589.
- [Mir+13] GR Mirams, CJ Arthurs, MO Bernabeu, R Bordas et al. ‘Chaste: An Open Source C++ Library for Computational Physiology and Biology’. In: *PLoS Computational Biology* 9.3 (2013), e1002970. DOI: [10.1371/journal.pcbi.1002970](https://doi.org/10.1371/journal.pcbi.1002970).
- [Mis+16] G Misirli, M Cavaliere, W Waites, M Pocock et al. ‘Annotation of Rule-Based Models with Formal Semantics to Enable Creation, Analysis, Reuse and Visualization’. In: *Bioinformatics* 32.6 (15 March 2016), pages 908–917. ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btv660](https://doi.org/10.1093/bioinformatics/btv660).
- [MLS13] D McGuinness, T Lebo and S Sahoo. ‘PROV-O: The PROV Ontology’. W3C Recommendation. W3C, April 2013. URL: <http://www.w3.org/TR/2013/REC-prov-o-20130430/>.
- [MN13] M Mattioni and NL Novère. ‘Integration of Biochemical and Electrical Signaling-Multiscale Model of the Medium Spiny Neuron of the Striatum’. In: *PLOS ONE* 8.7 (3 July 2013), e66811. ISSN: 1932-6203. DOI: [10.1371/journal.pone.0066811](https://doi.org/10.1371/journal.pone.0066811).
- [Moo+15] S Moodie, N Le Novère, E Demir, H Mi et al. ‘Systems Biology Graphical Notation: Process Description Language Level 1 Version 1.3’. In: *Journal of Integrative Bioinformatics* (2015). DOI: [10.2390/biecoll-jib-2015-263](https://doi.org/10.2390/biecoll-jib-2015-263).
- [Mor+08] L Moreau, P Groth, S Miles, J Vazquez-Salceda et al. ‘The Provenance of Electronic Data’. In: *Commun. ACM* 51.4 (April 2008), pages 52–58. ISSN: 0001-0782. DOI: [10.1145/1330311.1330323](https://doi.org/10.1145/1330311.1330323).
- [Mul11] A Mullard. ‘Reliability of ‘New Drug Target’ Claims Called into Question’. In: *Nature Reviews Drug Discovery* 10.9 (September 2011), pages 643–644. ISSN: 1474-1776. DOI: [10.1038/nrd3545](https://doi.org/10.1038/nrd3545).
- [Mye86] EW Myers. ‘An O(ND) Difference Algorithm and Its Variations’. In: *Algorithmica* 1 (1-4 1986), pages 251–266. ISSN: 0178-4617. DOI: [10.1007/BF01840446](https://doi.org/10.1007/BF01840446).
- [NB08] D Nickerson and M Buist. ‘Practical Application of CellML 1.1: The Integration of New Mechanisms into a Human Ventricular Myocyte Model’. In: *Progress in Biophysics and Molecular Biology* 98.1 (September 2008), pages 38–51. ISSN: 0079-6107. DOI: [10.1016/j.pbiomolbio.2008.05.006](https://doi.org/10.1016/j.pbiomolbio.2008.05.006).
- [NCB16] NCBI Resource Coordinators. ‘Database Resources of the National Center for Biotechnology Information’. In: *Nucleic Acids Research* 44 (Database issue 4 January 2016), pages D7–D19. ISSN: 0305-1048. DOI: [10.1093/nar/gkv1290](https://doi.org/10.1093/nar/gkv1290). PMID: 26615191.

Bibliography

- [Nie+09] SA Niederer, M Fink, D Noble and NP Smith. ‘A Meta-Analysis of Cardiac Electrophysiology Computational Models’. In: *Experimental Physiology* 94.5 (May 2009), pages 486–495. ISSN: 1469-445X. DOI: 10.1113/expphysiol.2008.044610. PMID: 19139063.
- [Nil+08] M Nilsson, A Powell, P Johnston and A Naeve. ‘Expressing Dublin Core Metadata Using the Resource Description Framework (RDF)’. Dublin Core Metadata Initiative, 14 January 2008. URL: <http://dublincore.org/documents/dc-rdf/> (accessed 3 April 2017).
- [Nob08] D Noble. ‘The Music of Life: Biology Beyond Genes’. Published: Paperback. Oxford University Press, USA, 2008. ISBN: 0-19-922836-1. URL: <https://global.oup.com/academic/product/the-music-of-life-9780199295739>.
- [Nob11] D Noble. ‘Successes and Failures in Modeling Heart Cell Electrophysiology’. In: *Heart Rhythm* 8.11 (November 2011), pages 1798–1803. ISSN: 15475271. DOI: 10.1016/j.hrthm.2011.06.014.
- [Nob60] D Noble. ‘Cardiac Action and Pacemaker Potentials Based on the Hodgkin-Huxley Equations’. In: *Nature* 188.4749 (5 November 1960), pages 495–497. ISSN: 0028-0836. DOI: 10.1038/188495b0.
- [Nos+15] BA Nosek, G Alter, GC Banks, D Borsboom et al. ‘Promoting an Open Research Culture’. In: *Science* 348.6242 (26 June 2015), pages 1422–1425. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aab2374. PMID: 26113702.
- [Nov+05] NL Novere, A Finney, M Hucka, US Bhalla et al. ‘Minimum Information Requested in the Annotation of Biochemical Models (MIRIAM)’. In: *Nat. Biotechnol.* 23.12 (December 2005), pages 1509–1515. DOI: 10.1038/nbt1156.
- [Nov+09] NL Novere, M Hucka, H Mi, S Moodie et al. ‘The Systems Biology Graphical Notation’. In: *Nat. Biotechnol.* 27.8 (August 2009), pages 735–741. DOI: 10.1038/nbt.1558.
- [Noy+03] NF Noy, M Crubézy, RW Fergerson, H Knublauch et al. ‘Protégé-2000: An Open-Source Ontology-Development and Knowledge-Acquisition Environment’. In: *AMIA Annual Symposium Proceedings 2003* (2003), page 953. ISSN: 1942-597X. PMID: 14728458. URL: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1480139/>.
- [NR01] D Noble and Y Rudy. ‘Models of Cardiac Ventricular Action Potentials: Iterative Interaction between Experiment and Simulation’. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 359.1783 (15 June 2001), pages 1127–1142. ISSN: 1364-503X, 1471-2962. DOI: 10.1098/rsta.2001.0820.

- [NT93] B Novak and JJ Tyson. ‘Numerical Analysis of a Comprehensive Model of M-Phase Control in *Xenopus* Oocyte Extracts and Intact Embryos.’ In: *Journal of Cell Science* 106.4 (December 1993), pages 1153–1168. ISSN: 0021-9533. PMID: 8126097. URL: <http://jcs.biologists.org/content/106/4/1153.abstract>.
- [OB15] BG Olivier and FT Bergmann. ‘SBML Level 3 Flux Balance Constraints Package Version 2 Release 1’. SBML Community, 12 September 2015. URL: <http://co.mbine.org/specifications/sbml.level-3.version-1.fbc.version-2.release-1> (accessed 20 April 2017).
- [OHa+11] T O’Hara, L Virág, A Varró and Y Rudy. ‘Simulation of the Undiseased Human Cardiac Ventricular Action Potential: Model Formulation and Experimental Validation’. In: *PLOS Computational Biology* 7.5 (26 May 2011), e1002061. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1002061.
- [OR12] T O’Hara and Y Rudy. ‘Quantitative Comparison of Cardiac Ventricular Myocyte Electrophysiology and Response to Drugs in Human and Nonhuman Species’. In: *American Journal of Physiology - Heart and Circulatory Physiology* 302.5 (1 March 2012), H1023–H1030. ISSN: 0363-6135, 1522-1539. DOI: 10.1152/ajpheart.00785.2011. PMID: 22159993.
- [OS04] BG Olivier and JL Snoep. ‘Web-Based Kinetic Modelling Using JWS Online’. In: *Bioinformatics* 20.13 (April 2004), pages 2143–2144. DOI: 10.1093/bioinformatics/bth200.
- [OZB08] GM Olson, A Zimmerman and N Bos. ‘Scientific Collaboration on the Internet’. The MIT Press, 2008. ISBN: 978-0-262-15120-7.
- [Pat12] D Pattinson. ‘PLOS ONE Launches Reproducibility Initiative’. 14 August 2012. URL: <https://s.binfalse.de/plosrepro> (accessed 3 April 2017).
- [Pen11] RD Peng. In: *Science* 334.6060 (December 2011), pages 1226–1227. DOI: 10.1126/science.1213847.
- [Per11] S Perreault. ‘vCard Format Specification’. IETF Trust, 2011. URL: <https://tools.ietf.org/html/rfc6350> (accessed 3 April 2017).
- [Pet16] M Peters. ‘Storage Solution for Models with Multiple Versions in Systems Biology: Concept and Prototype Implementation’. Bachelorarbeit. Rostock: University of Rostock, 15 November 2016. 75 pages. URL: <https://github.com/FreakyBytes/bachelor-thesis>.
- [PEU14] D Peng, R Ewald and AM Uhrmacher. ‘Towards Semantic Model Composition via Experiments’. In: *Proceedings of the 2Nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. SIGSIM PADS ’14. New York, NY, USA: ACM,

Bibliography

- 2014, pages 151–162. ISBN: 978-1-4503-2794-7. DOI: 10.1145/2601381.2601394. URL: <http://doi.acm.org/10.1145/2601381.2601394>.
- [PG14] P Pathmanathan and RA Gray. ‘Verification of Computational Models of Cardiac Electro-Physiology’. In: *International Journal for Numerical Methods in Biomedical Engineering* 30.5 (May 2014), pages 525–544. ISSN: 2040-7947. DOI: 10.1002/cnm.2615. PMID: 24259465.
- [Pit+09] J Pitt-Francis, P Pathmanathan, MO Bernabeu, R Bordas et al. ‘Chaste: A Test-Driven Approach to Software Development for Biological Modelling’. In: *Computer Physics Communications* 180.12 (December 2009), pages 2452–2471. ISSN: 00104655. DOI: 10.1016/j.cpc.2009.07.019.
- [Pok+12] A Pokhilko, AP Fernández, KD Edwards, MM Southern et al. ‘The Clock Gene Circuit in *Arabidopsis* Includes a Repressilator with Additional Feedback Loops’. In: *Molecular Systems Biology* 8.1 (1 January 2012), page 574. ISSN: 1744-4292, 1744-4292. DOI: 10.1038/msb.2012.6. PMID: 22395476.
- [Pot15] C Pottage. ‘A Comparison of XML Differencing Techniques on Systems Biology Markup Language (SBML) Datasets’. Project Report. Manchester: University of Manchester, April 2015.
- [PSA11] F Prinz, T Schlange and K Asadullah. ‘Believe It or Not: How Much Can We Rely on Published Data on Potential Drug Targets?’ In: *Nature Reviews Drug Discovery* 10.9 (September 2011), pages 712–712. ISSN: 1474-1776. DOI: 10.1038/nrd3439-c1.
- [Qu+14] Z Qu, G Hu, A Garfinkel and JN Weiss. ‘Nonlinear and Stochastic Dynamics in the Heart’. In: *Physics Reports* 543.2 (10 October 2014), pages 61–162. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2014.05.002. PMID: 25267872.
- [Qui+11] TA Quinn, S Granite, MA Allesie, C Antzelevitch et al. ‘Minimum Information about a Cardiac Electrophysiology Experiment (MICEE): Standardised Reporting for Model Reproducibility, Interoperability, and Data Sharing’. In: *Progress in Biophysics and Molecular Biology* 107.1 (October 2011), pages 4–10. ISSN: 1873-1732. DOI: 10.1016/j.pbiomolbio.2011.07.001. PMID: 21745496.
- [RDF14] RDF Working Group. ‘RDF - Semantic Web Standards’. 25 February 2014. URL: <https://www.w3.org/RDF/> (accessed 3 April 2017).
- [Rei91] C Reichenberger. ‘Delta Storage for Arbitrary Non-Text Files’. In: *Proceedings of the 3rd International Workshop on Software Configuration Management*. SCM ’91. New York, NY, USA: ACM, 1991, pages 144–152. ISBN: 978-0-89791-429-1. DOI: 10.1145/111062.111081. URL: <http://doi.acm.org/10.1145/111062.111081>.

- [RL10] S Ram and J Liu. ‘Provenance Management in BioSciences.’ In: *ER Workshops*. Edited by J Trujillo, G Dobbie, H Kangassalo, S Hartmann et al. Volume 6413. Lecture Notes in Computer Science. Springer, 2010, pages 54–64. ISBN: 978-3-642-16384-5. URL: <http://dblp.uni-trier.de/db/conf/er/erw2010.html#RamL10>.
- [RM03] C Rosse and JLV Mejino Jr. ‘A Reference Ontology for Biomedical Informatics: The Foundational Model of Anatomy’. In: *Journal of Biomedical Informatics*. Unified Medical Language System 36.6 (December 2003), pages 478–500. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2003.11.007.
- [Roh+12] H Rohn, A Junker, A Hartmann, E Grafahrend-Belau et al. ‘VANTED v2: A Framework for Systems Biology Applications’. In: *BMC Systems Biology* 6 (2012), page 139. ISSN: 1752-0509. DOI: 10.1186/1752-0509-6-139.
- [Rou+10] DD Roure, C Goble, S Aleksejevs, S Bechhofer et al. ‘Towards Open Science: The myExperiment Approach’. In: *Concurrency and Computation: Practice and Experience* 22.17 (2010), pages 2335–2353. DOI: 10.1002/cpe.1601.
- [RSB05] S Rönnaau, J Scheffczyk and UM Borghoff. ‘Towards XML Version Control of Office Documents’. In: *Proceedings of the 2005 ACM Symposium on Document Engineering*. DocEng ’05. New York, NY, USA: ACM, 2005, pages 10–19. ISBN: 1-59593-240-2. DOI: 10.1145/1096601.1096606. URL: <http://doi.acm.org/10.1145/1096601.1096606>.
- [RWE13] I Robinson, J Webber and E Eifrem. ‘Graph Databases’. Edited by O’Reilly. O’Reilly, 2013. ISBN: 978-1-4493-5626-2.
- [Sag+14] PT Sager, G Gintant, JR Turner, S Pettit et al. ‘Rechanneling the Cardiac Proarrhythmia Safety Paradigm: A Meeting Report from the Cardiac Safety Research Consortium’. In: *American Heart Journal* 167.3 (March 2014), pages 292–300. ISSN: 1097-6744. DOI: 10.1016/j.ahj.2013.11.004. PMID: 24576511.
- [San+08] SA Sansone, P Rocca-Serra, M Brandizi, A Brazma et al. ‘The First RSBI (ISATAB) Workshop: “Can a Simple Format Work for Complex Studies?”’ In: *OMICS: A Journal of Integrative Biology* 12.2 (30 April 2008), pages 143–149. ISSN: 1536-2310. DOI: 10.1089/omi.2008.0019.
- [San+12] SA Sansone, P Rocca-Serra, D Field, E Maguire et al. ‘Toward Interoperable Bioscience Data’. In: *Nature Genetics* 44.2 (February 2012), pages 121–126. ISSN: 1061-4036. DOI: 10.1038/ng.1054.
- [San+13] GK Sandve, A Nekrutenko, J Taylor and E Hovig. ‘Ten Simple Rules for Reproducible Computational Research’. In: *PLoS Comput. Biol.* 9.10 (October 2013), e1003285. DOI: 10.1371/journal.pcbi.1003285.

Bibliography

- [Sau+16] HM Sauro, K Choi, JK Medley, C Cannistra et al. ‘Tellurium: A Python Based Modeling and Reproducibility Platform for Systems Biology’. In: *bioRxiv* (21 May 2016), page 054601. DOI: [10.1101/054601](https://doi.org/10.1101/054601).
- [SBM17] SBML Editors. ‘SBML Software Guide’. 2017. URL: http://sbml.org/SBML_Software_Guide (accessed 20 April 2017).
- [SBS10] SY Shen, F Bergmann and HM Sauro. ‘SBML2TikZ: Supporting the SBML Render Extension in LaTeX’. In: *Bioinformatics* 26.21 (2010), pages 2794–2795. DOI: [10.1093/bioinformatics/btq512](https://doi.org/10.1093/bioinformatics/btq512).
- [Sch+14a] M Scharm, F Wendland, M Peters, M Wolfien et al. ‘The CombineArchive Toolkit - Facilitating the Transfer of Research Results’. In: *PeerJ Preprints* (2014). DOI: [10.7287/peerj.preprints.514v1](https://doi.org/10.7287/peerj.preprints.514v1).
- [Sch+14b] M Scharm, F Wendland, M Peters, M Wolfien et al. ‘The CombineArchiveWeb Application – A Web Based Tool to Handle Files Associated with Modelling Results’. In: *Proceedings of the 2014 Workshop on Semantic Web Applications and Tools for Life Sciences*. SWAT4LS 2014. Berlin, GER, 2014. DOI: [10.7287/peerj.preprints.639v1](https://doi.org/10.7287/peerj.preprints.639v1). URL: <http://ceur-ws.org/Vol-1320/>.
- [Sch+14c] F Schliess, S Hoehme, SG Henkel, A Ghallab et al. ‘Integrated Metabolic Spatial-Temporal Model for the Prediction of Ammonia Detoxification during Liver Damage and Regeneration’. In: *Hepatology* 60.6 (1 December 2014), pages 2040–2051. ISSN: [1527-3350](https://doi.org/10.1002/hep.27136). DOI: [10.1002/hep.27136](https://doi.org/10.1002/hep.27136).
- [Sch+16] M Scharm, D Waltemath, P Mendes and O Wolkenhauer. ‘COMODI: An Ontology to Characterise Differences in Versions of Computational Models in Biology’. In: *Journal of Biomedical Semantics* 7 (2016), page 46. ISSN: [2041-1480](https://doi.org/10.1186/s13326-016-0080-2). DOI: [10.1186/s13326-016-0080-2](https://doi.org/10.1186/s13326-016-0080-2).
- [Sch+18] M Scharm, T Gebhardt, V Touré, A Bagnacani et al. ‘Evolution of Computational Models in BioModels Database and the Physiome Model Repository’. In: *BMC Systems Biology* 12 (12 April 2018), page 53. ISSN: [1752-0509](https://doi.org/10.1186/s12918-018-0553-2). DOI: [10.1186/s12918-018-0553-2](https://doi.org/10.1186/s12918-018-0553-2).
- [Sch13] M Scharm. ‘Improving the Management of Computational Models’. Invited Talk. Invited Talk at the European Bioinformatics Institute. Hinxton, Cambridgeshire, UK, 19 December 2013. URL: <https://www.slideshare.net/binfalse/martin-scharmebidec2013> (accessed 4 July 2017).
- [Sha05] Y Shafranovich. ‘Common Format and MIME Type for Comma-Separated Values (CSV) Files’. The Internet Society, 2005. URL: <https://tools.ietf.org/html/rfc4180> (accessed 24 April 2017).

- [SHL07] L Strömbäck, D Hall and P Lambrix. ‘A Review of Standards for Data Exchange within Systems Biology’. In: *PROTEOMICS* 7.6 (1 March 2007), pages 857–867. ISSN: 1615-9861. DOI: [10.1002/pmic.200600438](https://doi.org/10.1002/pmic.200600438).
- [Smi+07] B Smith, M Ashburner, C Rosse, J Bard et al. ‘The OBO Foundry: Coordinated Evolution of Ontologies to Support Biomedical Data Integration’. In: *Nat Biotechnol* 25.11 (November 2007), pages 1251–1255. DOI: [10.1038/nbt1346](https://doi.org/10.1038/nbt1346).
- [Smi+13] LP Smith, M Hucka, S Hoops, A Finney et al. ‘SBML Level 3 Hierarchical Model Composition Package Version 1 Release 3’. 2013. URL: <http://identifiers.org/combine.specifications/sbml.level-3.version-1.comp.version-1.release-3> (accessed 2 April 2017).
- [Smi+15] LP Smith, M Hucka, S Hoops, A Finney et al. ‘SBML Level 3 Package: Hierarchical Model Composition, Version 1 Release 3’. In: *Journal of Integrative Bioinformatics* (2015). DOI: [10.2390/biecoll-jib-2015-268](https://doi.org/10.2390/biecoll-jib-2015-268).
- [SO09] P Saffrey and R Orton. ‘Version Control of Pathway Models Using XML Patches’. In: *BMC Syst Biol* 3 (2009), page 34. DOI: [10.1186/1752-0509-3-34](https://doi.org/10.1186/1752-0509-3-34).
- [Sob09] EA Sobie. ‘Parameter Sensitivity Analysis in Electrophysiological Models Using Multivariable Regression’. In: *Biophysical Journal* 96.4 (18 February 2009), pages 1264–1274. ISSN: 0006-3495. DOI: [10.1016/j.bpj.2008.10.056](https://doi.org/10.1016/j.bpj.2008.10.056). PMID: 19217846.
- [Sor+15] A Sorokin, N Le Novère, A Luna, T Czauderna et al. ‘Systems Biology Graphical Notation: Entity Relationship Language Level 1 Version 2’. In: *Journal of Integrative Bioinformatics* (2015). DOI: [10.2390/biecoll-jib-2015-264](https://doi.org/10.2390/biecoll-jib-2015-264).
- [SP16] J Scott-Brown and A Papachristodoulou. ‘Sbml-Diff: A Tool for Visually Comparing SBML Models in Synthetic Biology’. In: *ACS Synthetic Biology* (30 December 2016). DOI: [10.1021/acssynbio.6b00273](https://doi.org/10.1021/acssynbio.6b00273).
- [ST16] M Scharm and V Touré. ‘COMBINE Archive Show Case’. 10 June 2016. DOI: [10.6084/m9.figshare.3427271.v1](https://doi.org/10.6084/m9.figshare.3427271.v1). URL: https://figshare.com/articles/COMBINE_Archive_Show_Case/3427271.
- [Sta+15] NJ Stanford, K Wolstencroft, M Golebiewski, R Kania et al. ‘The Evolution of Standards and Data Management Practices in Systems Biology’. In: *Molecular Systems Biology* 11.12 (1 December 2015), page 851. ISSN: 1744-4292, 1744-4292. DOI: [10.15252/msb.20156053](https://doi.org/10.15252/msb.20156053). PMID: 26700851.
- [SW15] M Scharm and D Waltemath. ‘Extracting Reproducible Simulation Studies from Model Repositories Using the CombineArchive Toolkit’. In: *Datenbanksysteme Für Business, Technologie Und Web (BTW 2015)*. Edited by W Wingerath, A Henrich,

Bibliography

- W Lehner, A Thor et al. 2015. URL: <http://subs.emis.de/LNI/Proceedings/Proceedings242.html>.
- [SW16a] M Scharm and D Waltemath. 'A Fully Featured COMBINE Archive of a Simulation Study on Syncytial Mitotic Cycles in Drosophila Embryos'. In: *F1000Research* 5 (29 September 2016), page 2421. ISSN: 2046-1402. DOI: 10.12688/f1000research.9379.1.
- [SW16b] M Scharm and D Waltemath. 'Model Management in Systems Biology: Challenges – Approaches – Solutions'. Webinar. Webinar. 2016. URL: <https://www.slideshare.net/binfalse/model-management-in-systems-biology-challenges-approaches-solutions> (accessed 21 March 2017).
- [Swa+15] M Swat, S Moodie, S Wimalaratne, N Kristensen et al. 'Pharmacometrics Markup Language (PharmML): Opening New Perspectives for Model Exchange in Drug Development: PharmML - Pharmacometrics Markup Language'. In: *CPT: Pharmacometrics & Systems Pharmacology* 4.6 (June 2015), pages 316–319. ISSN: 21638306. DOI: 10.1002/psp4.57.
- [SWW14] M Scharm, O Wolkenhauer and D Waltemath. 'Identifying, Interpreting, and Communicating Changes in XML-Encoded Models of Biological Systems'. In: 10th International Conference on Data Integration in the Life Sciences. Lisbon, 2014. URL: <http://dils2014.inesc-id.pt/>.
- [SWW16] M Scharm, O Wolkenhauer and D Waltemath. 'An Algorithm to Detect and Communicate the Differences in Computational Models Describing Biological Systems'. In: *Bioinformatics* 32.4 (15 February 2016), pages 563–570. ISSN: 1367-4803, 1460-2059. DOI: 10.1093/bioinformatics/btv484.
- [Tay+07] CF Taylor, NW Paton, KS Lilley, PA Binz et al. 'The Minimum Information about a Proteomics Experiment (MIAPE)'. In: *Nature Biotechnology* 25.8 (August 2007), pages 887–893. ISSN: 1087-0156. DOI: 10.1038/nbt1329.
- [Tay+08] CF Taylor, D Field, SA Sansone, J Aerts et al. 'Promoting Coherent Minimum Reporting Guidelines for Biological and Biomedical Investigations: The MIBBI Project'. In: *Nature Biotechnology* 26.8 (August 2008), pages 889–896. ISSN: 1087-0156. DOI: 10.1038/nbt0808-889.
- [Ten+06] KHWJ Ten Tusscher, O Bernus, R Hren and AV Panfilov. 'Comparison of Electrophysiological Models for Human Ventricular Cells and Tissues'. In: *Progress in Biophysics and Molecular Biology* 90 (1-3 2006), pages 326–345. ISSN: 0079-6107. DOI: 10.1016/j.pbiomolbio.2005.05.015. PMID: 16002127.

- [The07] The UniProt Consortium. ‘The Universal Protein Resource (UniProt)’. In: *Nucleic Acids Research* 35 (Database issue January 2007), pages D193–D197. ISSN: 0305-1048. DOI: [10.1093/nar/gkl929](https://doi.org/10.1093/nar/gkl929). PMID: 17142230.
- [The13] The Economist editors. ‘How Science Goes Wrong’. 21 October 2013. URL: <http://www.economist.com/news/leaders/21588069-scientific-research-has-changed-world-now-it-needs-change-itself-how-science-goes-wrong> (accessed 24 March 2017).
- [Thi+13] I Thiele, N Swainston, RMT Fleming, A Hoppe et al. ‘A Community-Driven Global Reconstruction of Human Metabolism’. In: *Nature Biotechnology* 31.5 (May 2013), pages 419–425. ISSN: 1087-0156. DOI: [10.1038/nbt.2488](https://doi.org/10.1038/nbt.2488).
- [Tic85] WF Tichy. ‘RCS — a System for Version Control’. In: *Software: Practice and Experience* 15.7 (1985), pages 637–654. ISSN: 1097-024X. DOI: [10.1002/spe.4380150703](https://doi.org/10.1002/spe.4380150703).
- [Top+14] M Topalidou, A Leblois, T Boraud and NP Rougier. ‘A Long Journey into Reproducible Computational Neuroscience Research’. Poster Published: Fourth International Symposium on Biology of Decision Making (SBDM 2014). May 2014. URL: <https://hal.inria.fr/hal-01090648>.
- [Tra11] NA Trayanova. ‘Whole-Heart Modeling’. In: *Circulation Research* 108.1 (7 January 2011), pages 113–128. ISSN: 0009-7330, 1524-4571. DOI: [10.1161/CIRCRESAHA.110.223610](https://doi.org/10.1161/CIRCRESAHA.110.223610). PMID: 21212393.
- [Tys91] JJ Tyson. ‘Modeling the Cell Division Cycle: Cdc2 and Cyclin Interactions.’ In: *Proceedings of the National Academy of Sciences of the United States of America* 88.16 (15 August 1991), pages 7328–7332. ISSN: 0027-8424. PMID: 1831270. URL: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC52288/>.
- [vGen+07] C van Gend, R Conradie, D Preez, F B et al. ‘Data and Model Integration Using JWS Online’. In: *In Silico Biology* 7 (2 Supplement 1 January 2007), pages 27–35. ISSN: 1386-6338. URL: <http://content.iospress.com/articles/in-silico-biology/isb00303>.
- [Wal+11a] D Waltemath, R Adams, DA Beard, FT Bergmann et al. ‘Minimum Information About a Simulation Experiment (MIASE)’. In: *PLOS Computational Biology* 7.4 (28 April 2011), e1001122. ISSN: 1553-7358. DOI: [10.1371/journal.pcbi.1001122](https://doi.org/10.1371/journal.pcbi.1001122).
- [Wal+11b] D Waltemath, R Adams, F Bergmann, M Hucka et al. ‘Reproducible Computational Biology Experiments with SED-ML – the Simulation Experiment Description Markup Language’. In: *BMC Systems Biology* 5.1 (2011), page 198. ISSN: 1752-0509. DOI: [10.1186/1752-0509-5-198](https://doi.org/10.1186/1752-0509-5-198).

Bibliography

- [Wal+11c] D Waltemath, F Bergmann, R Adams and N Le Novère. ‘Simulation Experiment Description Markup Language (SED-ML) : Level 1 Version 1’. In: *Nature Precedings* (25 March 2011). ISSN: 1756-0357. DOI: [10.1038/npre.2011.5846.1](https://doi.org/10.1038/npre.2011.5846.1).
- [Wal+13] D Waltemath, R Henkel, R Hälke, M Scharm et al. ‘Improving the Reuse of Computational Models through Version Control’. In: *Bioinformatics* 29.6 (2013), pages 742–748. DOI: [10.1093/bioinformatics/btt018](https://doi.org/10.1093/bioinformatics/btt018).
- [Wal+16] D Waltemath, R Henkel, R Hoehndorf, T Kacprowski et al. ‘Notions of Similarity for Computational Biology Models’. In: *bioRxiv* (21 March 2016), page 044818. DOI: [10.1101/044818](https://doi.org/10.1101/044818).
- [Whe+11] PL Whetzel, NF Noy, NH Shah, PR Alexander et al. ‘BioPortal: Enhanced Functionality via New Web Services from the National Center for Biomedical Ontology to Access and Use Ontologies in Software Applications’. In: *Nucleic Acids Research* 39 (June 2011), W541–W545. DOI: [10.1093/nar/gkr469](https://doi.org/10.1093/nar/gkr469).
- [Wil+16] MD Wilkinson, M Dumontier, IJ Aalbersberg, G Appleton et al. ‘The FAIR Guiding Principles for Scientific Data Management and Stewardship’. In: *Scientific Data* 3 (15 March 2016). ISSN: 2052-4463. DOI: [10.1038/sdata.2016.18](https://doi.org/10.1038/sdata.2016.18). PMID: 26978244.
- [Wim+09a] SM Wimalaratne, MDB Halstead, CM Lloyd, MT Cooling et al. ‘A Method for Visualizing CellML Models’. In: *Bioinformatics* 25.22 (2009), pages 3012–3019. DOI: [10.1093/bioinformatics/btp495](https://doi.org/10.1093/bioinformatics/btp495).
- [Wim+09b] SM Wimalaratne, MDB Halstead, CM Lloyd, MT Cooling et al. ‘Facilitating Modularity and Reuse: Guidelines for Structuring CellML 1.1 Models by Isolating Common Biophysical Concepts’. In: *Experimental Physiology* 94.5 (1 May 2009), pages 472–485. ISSN: 1469-445X. DOI: [10.1113/expphysiol.2008.045161](https://doi.org/10.1113/expphysiol.2008.045161).
- [Wim+09c] SM Wimalaratne, MDB Halstead, CM Lloyd, EJ Crampin et al. ‘Biophysical Annotation and Representation of CellML Models’. In: *Bioinformatics* 25.17 (1 September 2009), pages 2263–2270. ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btp391](https://doi.org/10.1093/bioinformatics/btp391).
- [Wol+11] K Wolstencroft, S Owen, F du Preez, O Krebs et al. ‘The SEEK: A Platform for Sharing Data and Models in Systems Biology’. In: *Meth. Enzymol.* 500 (2011), pages 629–655. DOI: [10.1016/B978-0-12-385118-5.00029-3](https://doi.org/10.1016/B978-0-12-385118-5.00029-3).
- [Wol+15] K Wolstencroft, S Owen, O Krebs, Q Nguyen et al. ‘SEEK: A Systems Biology Data and Model Management Platform’. In: *BMC Systems Biology* 9.1 (July 2015). DOI: [10.1186/s12918-015-0174-y](https://doi.org/10.1186/s12918-015-0174-y).
- [Wol+17] K Wolstencroft, O Krebs, JL Snoep, NJ Stanford et al. ‘FAIRDOMHub: A Repository and Collaboration Environment for Sharing Systems Biology Research’. In: *Nucleic Acids Research* 45 (D1 4 January 2017), pages D404–D407. ISSN: 0305-1048. DOI: [10.1093/nar/gkw1032](https://doi.org/10.1093/nar/gkw1032).

- [WW16] D Waltemath and O Wolkenhauer. ‘How Modeling Standards, Software, and Initiatives Support Reproducibility in Systems Biology and Systems Medicine’. In: *IEEE Transactions on Biomedical Engineering* 63.10 (October 2016), pages 1999–2006. ISSN: 0018-9294, 1558-2531. DOI: 10.1109/TBME.2016.2555481.
- [WW97] M Wolf and C Wicksteed. ‘Date and Time Formats’. w3c, 1997. URL: <https://www.w3.org/TR/NOTE-datetime> (accessed 3 April 2017).
- [Yu+11] T Yu, CM Lloyd, DP Nickerson, MT Cooling et al. ‘The Physiome Model Repository 2’. In: *Bioinformatics* 27.5 (1 March 2011), pages 743–744. ISSN: 1367-4803, 1460-2059. DOI: 10.1093/bioinformatics/btq723.
- [Zhu+12] A Zhukova, R Adams, C Laibe and N Le Novère. ‘LibKiSAO: A Java Library for Querying KiSAO’. In: *BMC Research Notes* 5 (2012), page 520. ISSN: 1756-0500. DOI: 10.1186/1756-0500-5-520.

Appendices

APPENDIX A

ISSUES WITH LCS AND COMPUTATIONAL MODELS

As described in Section 1.2.3 and discussed in [Wal+13], solutions for difference detection before this thesis were typically based on Unix' diff, i.e. effectively relying on the longest common subsequence (LCS) problem [HM76]. While this line-based approach is successfully applied to manage source code, it performs poorly for difference detection of models. The main reason is that LCS does not respect the XML structure [RSB05], which constitutes model files. The XML encoding may already change when a model is loaded and immediately exported (without any modifications) by a model editor. Every software tool has its own preferred way of representing the model code, sorting the occurring XML elements or breaking lines in the XML code. Such common changes are detected by the LCS algorithm, but they are in fact irrelevant for the model's history and would be neglected by entity-based algorithms.

When studying the evolution of open model repositories (Section 3.4) I also analysed how my novel algorithm (Section 2.2) performs in comparison to Unix' diff. Table A.1 compares the performance of both tools in terms of delta composition. As soon as a single character (including white space) in a model has changed it is reported as a change by Unix' diff. Thus,

Number of deltas	12 467
Empty deltas reported by Unix' diff	0
Empty deltas reported by BiVeS	736
Average number of operations reported by Unix' diff	5453.01
Average number of operations reported by BiVeS	348.49

Table A.1. Comparison of BiVeS and Unix' diff. The table shows figures on the performance of BiVeS and Unix' diff during my large-scale study of the evolution of models in open repositories (see Section 3.4).

Appendix A. Issues with LCS and computational models

the number of empty deltas reported by Unix' diff is equal to zero. In contrast, BiVeS respects the hierarchical structure of model documents and neglects indentation, line breaks, or the order of arguments in an XML node. Therefore, it reports 736 deltas to contain no relevant changes. For the same reason, both tools report a significantly different number of operations which are necessary to transfer one version into the other. While Unix' diff needs on average more than 5000 operations, the BiVeS tool only reports about 350 operations. Moreover, please keep in mind, that BiVeS typically reports changes more verbose than Unix' diff. For example, a typical species in an SBML document looks like the following:

```
1 <species metaid="metaid_0000007" id="ATP" name="ATP_ADP_AMP_Ado" compartment="cell"  
  ↪ initialAmount="2475.35" hasOnlySubstanceUnits="true" />
```

If this species would be removed in a subsequent version, Unix' diff would report a single deletion, as this just affects one single line. In contrast, BiVeS would report seven differences: The deletion of a node tagged `species` and six deleted attributes. Consequently, one may intuitively expect a larger number of differences reported by BiVeS. However, the algorithm implemented in BiVeS still manages to outperform Unix' diff tool so clearly. The Appendix B contains another example illustrating the power of BiVeS with respect to mathematical equations.

APPENDIX B

BIVES EXAMPLE WITH MATHML

In this appendix I present a simple example demonstrating BiVeS' power and usefulness with respect to mathematical equations encoded in XML documents. Given the following two mathematical formulas:

$$3 \cdot 5 - 1 \cdot 7 \tag{B.1}$$

$$1 \cdot 7 - 3 \cdot 5 \tag{B.2}$$

For humans it may be obvious that the two multiplication terms (i.e. $3 \cdot 5$ and $1 \cdot 7$, respectively) are swapped between both formulas. This, in turn, affects the result: Equation B.1 evaluates to 8 and Equation B.2 results in -8 . However, there are of course other explanations possible. For example, the individual figures may have been replaced. That is, the 3 was removed and a 1 was inserted, the 5 was replaced for a 7, and so on. As the first explanation (swapping multiplication terms) requires less operations than the second explanation (replacing individual figures) it may be considered better, as it is easier to comprehend the change.

Standard formats based on XML typically use MathML to express mathematical equations in a machine-readable format. Encoded in MathML, above formulas seem a bit more complex, but the differences should still be graspable even for humans, see Source Code B.1. The default tool for file comparisons, Unix' diff, needs eight operations to transform the first version into the second version: Four operations deleted the original figures and four operations insert the new figures. The resulting patch of Unix' diff is shown in Figure B.1a. Similarly, ExamXML¹ reports eight entities that have changed, once it is configured to respect the ordering of siblings in an XML document. The result of ExamXML is shown in Figure B.1b.

¹www.a7soft.com/examxml.html, accessed 11 October 2017

Appendix B. BiVeS example with MathML

<pre> 1 <math> 2 <apply> 3 <minus/> 4 <apply> 5 <times/> 6 <cn>3</cn> 7 <cn>5</cn> 8 </apply> 9 <apply> 10 <times/> 11 <cn>1</cn> 12 <cn>7</cn> 13 </apply> 14 </apply> 15 </math> </pre>	<pre> 1 <math> 2 <apply> 3 <minus/> 4 <apply> 5 <times/> 6 <cn>1</cn> 7 <cn>7</cn> 8 </apply> 9 <apply> 10 <times/> 11 <cn>3</cn> 12 <cn>5</cn> 13 </apply> 14 </apply> 15 </math> </pre>
---	---

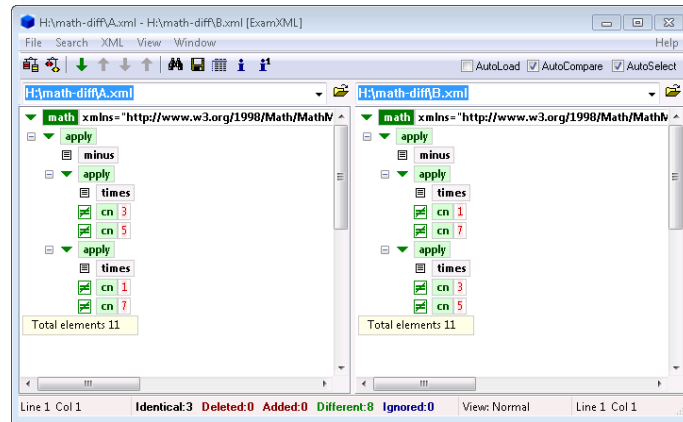
Source Code B.1. MathML encoding of simple formulas. The shown XML documents encode the formulas $3 \cdot 5 - 1 \cdot 7$ (left) and $1 \cdot 7 - 3 \cdot 5$ (right) in MathML. There are multiple possible explanations for the change. For example, the figures may have been replaced individually or the multiplication terms may have been swapped. The latter explains the change using less operations and may therefore be considered better in terms of tangibility.

```

1 6,7c6,7
2 < > > > <cn>3</cn>
3 < > > > <cn>5</cn>
4 - - -
5 > > > > <cn>1</cn>
6 > > > > <cn>7</cn>
7 11,12c11,12
8 < > > > <cn>1</cn>
9 < > > > <cn>7</cn>
10 - - -
11 > > > > <cn>3</cn>
12 > > > > <cn>5</cn>

```

(a) Result obtained by Unix' diff



(b) Comparison report in ExamXML

Figure B.1. Comparison of MathML using typical tools. Both Unix' diff (B.1a) and ExamXML (B.1b) detect eight changes between the two mathematical formulas encoded in MathML (as shown in Source Code B.1). To transfer one equation into the other they would delete all figures and insert the new ones.

In contrast, BiVeS is able to recognise that the multiplication terms were swapped. It does not report any insert or a delete, but detects the corresponding move operations. Source Code B.2 shows how BiVeS swaps children 2 and 3 of the topmost `apply` node. Thus, BiVeS is able to “explain” the changes using less operations.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bives type="fullDiff" id="bivesPatch">
3   <!--BiVeS compiled with: [BiVeS Framework v1.11.1] [BiVeS Core v1.8.3] [BiVeS SBML
   ↪ v1.8.2] [BiVeS CellML v1.7.3] -->
4   <update />
5   <delete />
6   <insert />
7   <move>
8     <node id="1" oldParent="/math[1]/apply[1]" newParent="/math[1]/apply[1]"
      ↪ oldChildNo="2" newChildNo="3" oldPath="/math[1]/apply[1]/apply[1]"
      ↪ newPath="/math[1]/apply[1]/apply[2]" />
9     <node id="2" oldParent="/math[1]/apply[1]" newParent="/math[1]/apply[1]"
      ↪ oldChildNo="3" newChildNo="2" oldPath="/math[1]/apply[1]/apply[2]"
      ↪ newPath="/math[1]/apply[1]/apply[1]" />
10  </move>
11 </bives>
```

Source Code B.2. MathML comparison result of BiVeS. When comparing the MathML trees shown in Source Code B.1, BiVeS is able to recognise that the multiplication terms have been swapped. It reports that children 2 and 3 (oldChildNo or newChildNo, respectively) of the node /math[1]/apply[1] need to be swapped.

APPENDIX C

BIVES' DELTA FORMAT

A typical delta reported by BiVeS is shown in Source Code C.1. Its root node `bives` is parent of four child nodes tagged (i) `update`, (ii) `delete`, (iii) `insert`, and (iv) `move`. Each of them may carry an unbounded number of *diff nodes* encoding for particular differences. There are three types of diff nodes according to the elements in the XML document affected by the modification:

- A diff node with a tag name `node` encodes changes on an XML document node. For example, line 13 of Source Code C.1 reveals that a document node was inserted to the modified version of the document.
- A diff node tagged `attribute` encodes changes on an attribute of an XML node. For example, lines 5 to 7 of Source Code C.1 show that three attributes were updated and line 10 informs that another attribute was deleted.
- A diff node with a tag name `text` encodes changes on a text node in the XML document. For example, line 14 of Source Code C.1 states that some text was added to the XML document.

The details of a certain difference are stored as attributes in the diff node. There are multiple possible attributes carrying different aspects of a modification, compare Table C.1. A special attribute `triggeredBy` marks differences that were triggered by other differences. For example, if a subtree of the XML document (e.g. a reaction) is deleted, then all nodes in the subtree (e.g. the kinetic law) are deleted as well. As BiVeS produces complete deltas, which can be used for transforming the documents into each other in both directions, the XML serialisation will contain a diff node for every atomic difference in the XML tree. Thus, even if the abstract change can be understood as a “deletion of a reaction”, the delta reported by BiVeS may contain hundreds of entities to represent this change. However, all nodes (except the node representing the deletion of the reaction) will carry an attribute `triggeredBy` pointing to the diff node that

Appendix C. BiVeS' delta format

entailed the operation. Similarly, if a document node is deleted all its attributes are removed as well. Every deleted attribute will therefore point to the deletion of the document node using a `triggeredBy` attribute. In addition to the attributes specifying a difference, every diff node carries a mandatory `id` attribute to unambiguously address the corresponding modification from, e.g., annotations. Since BiVeS version 1.4.6, the XML serialisation of the delta contains a comment stating the versions of BiVeS' modules used to create the delta, see line 3 of Source Code C.1.

A machine-readable XML schema definition is shipped with BiVeS' source code. It can, for example, be obtained from GitHub at s.binfalse.de/bivesxsd.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bives type="fullDiff" id="bivesPatch">
3   <!--BiVeS compiled with: [BiVeS Framework v1.9.2] [BiVeS Core v1.7.2] [BiVeS SBML
   ↪ v1.7.5] [BiVeS CellML v1.6.6] -->
4   <update>
5     <attribute name="name" id="1" oldValue="wolf_2001" newValue="wolf_2000"
   ↪   oldPath="/model[1]" newPath="/model[1]" />
6     <attribute name="id" id="2" oldValue="wolf_2001" newValue="wolf_2000"
   ↪   oldPath="/model[1]" newPath="/model[1]" />
7     <attribute name="about" id="4" oldValue="#wolf_2001" newValue="#wolf_2000"
   ↪   oldPath="/model[1]/RDF[1]/Description[19]"
   ↪   newPath="/model[1]/RDF[1]/Description[19]" />
8   </update>
9   <delete>
10    <attribute name="base" id="3" oldValue="" oldPath="/model[1]" />
11  </delete>
12  <insert>
13    <node id="1" newParent="/model[1]/RDF[1]/Description[2]/.../Bag[1]" newChildNo="3"
   ↪   newPath="/model[1]/RDF[1]/Description[2]/.../Bag[1]/li[3]" newTag="li" />
14    <text id="2" triggeredBy="1"
   ↪   newParent="/model[1]/RDF[1]/Description[2]/.../Bag[1]/li[3]" newChildNo="1"
   ↪   newPath="/model[1]/RDF[1]/Description[2]/.../Bag[1]/li[3]/text()[1]"
   ↪   newText="electrophysiology" />
15  </insert>
16  <move />
17 </bives>
```

Source Code C.1. Example of a serialised delta produced by BiVeS. The XML representation encodes all differences and can be used to reconstruct the original document given the modified version, and vice versa. The shown delta contains three updates of attribute values, the deletion of an attribute, and insertions of a document node and a text node.

Attribute	Applies to	Description
oldPath	node, attribute, text	path to the element in the original document; in case of changes on attributes it is the path to the document node carrying the corresponding attribute (XPath expression)
oldParent	node, text	path to the parent node in the original document hosting the affected node (XPath expression)
oldChildNo	node, text	child number among its siblings in the original document (integer)
oldTag	node	tag name of the node in the original document (string)
oldValue	attribute	attribute value in the original document (string)
oldText	text	textual content in the original document (string)
name	attribute	name of the attribute; please note that renaming an attribute is not supported (string)
newPath	node, attribute, text	path to the element in the modified document; in case of changes on attributes it is the path to the document node carrying the corresponding attribute (XPath expression)
newParent	node, text	path to the parent node in the modified document hosting the affected node (XPath expression)
newChildNo	node, text	child number among its siblings in the modified document (integer)
newTag	node	tag name of the node in the modified document (string)
newValue	attribute	attribute value in the modified document (string)
oldText	text	textual content in the modified document (string)
triggeredBy	node, attribute, text	holds the <code>id</code> of the diff node that triggered this operation (integer)

Table C.1. Attributes encoding for details of a modification in a diff node. The table lists possible attributes that carry details of a certain difference.

APPENDIX D

LIST OF SHORT URLS

As some URLs are very long it is difficult to record them in a paper based format. They sometimes leave the text block (e.g. <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/e/21730/21729/21750/21760/21716/21721/21718/21719/21720/21722/20061/20053/19270/19683/19674/20050/20049/show/1848458697/displayPlotFlot>) or break into parts and you never know if a dash belongs to a URL or not (e.g. <https://sems.uni-rostock.de/2016/06/a-container-for-the-combinearchive-web-interface/>). Both are challenging to read, destroy the layout, and nobody wants to type them anyway. Therefore I decided to use a URL-shortener service for very long URLs, unless I believe that the actual domain helps to understand the respective point. To shorten the URLs, I used the open source tool YOURLS¹, which is deployed at `s.binfalse.de`. Even though, the service is running stable at that end-point for more than seven years now, there is a risk that it will stop working properly at some point. Therefore, the following list collects the URL mappings used in this document, so that it is still possible to access the original URLs after my YOURLS instance deceased. Line breaks are introduced manually. That means, if a line ends in a hyphen there is a hyphen in the URL.

s.binfalse.de/AP1Hz <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/e/21715/21716/21721/21718/21719/21720/21722/20061/20053/19270/19683/19674/20050/20049/show/1848458697/displayPlotFlot>

s.binfalse.de/AP2Hz <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/e/21730/21729/21750/21763/21761/21760/21755/20062/20054/19274/19679/19675/20047/20046/show/1848458697/displayPlotFlot>

s.binfalse.de/bivesxsd <https://github.com/binfalse/BiVeS/blob/master/res/bives-diff-schema.xsd>

s.binfalse.de/ca-example <https://github.com/SemsProject/CombineArchive/blob/master/src/main/java/de/unirostock/sems/cbarchive/Example.java>

¹yourls.org, accessed 16 July 2017

Appendix D. List of short URLs

s.binfalse.de/CA1Hz <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/e/4737/4760/4530/4531/4532/4534/4535/4547/4548/4550/4554/4555/4556/4557/show/-977543105/displayPlotFlot>

s.binfalse.de/CA2Hz <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/e/5651/5652/5660/5661/5662/5663/5664/5669/5670/5647/5673/5674/5676/5677/show/-977543105/displayPlotFlot>

s.binfalse.de/cellmldiffannotator <https://github.com/binfalse/BiVeS-CellML/blob/master/src/main/java/de/unirostock/sems/bives/cellml/algorithm/CellMLDiffAnnotator.java>

s.binfalse.de/changefactory <https://github.com/SemsProject/jCOMODI/blob/master/src/main/java/de/unirostock/sems/comodi/ChangeFactory.java>

s.binfalse.de/decker-ap <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/e/5446/4526/5655/5520/5693/show/1848458697/displayPlotFlot>

s.binfalse.de/decker-buggy <https://travis.cs.ox.ac.uk/FunctionalCuration/model/Decker2009/53/20140428181215/89>

s.binfalse.de/decker-diff <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/m/90/89/show/-1805825569/displayBivesDiff>

s.binfalse.de/decker-fixed <https://travis.cs.ox.ac.uk/FunctionalCuration/model/Decker2009/53/fixed/90/>

s.binfalse.de/decker-s1s2 <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/e/5615/5407/show/-1767010368/displayPlotFlot>

s.binfalse.de/decker-versions <https://travis.cs.ox.ac.uk/FunctionalCuration/model/Decker2009/53/>

s.binfalse.de/defaultdiffannotator <https://github.com/binfalse/BiVeS-Core/blob/master/src/main/java/de/unirostock/sems/bives/algorithm/general/DefaultDiffAnnotator.java>

s.binfalse.de/diss-v1 <https://scratch.binfalse.de/diss/version1.xml>

s.binfalse.de/diss-v2 <https://scratch.binfalse.de/diss/version2.xml>

s.binfalse.de/ecma-376 <http://www.ecma-international.org/publications/standards/Ecma-376.htm>

s.binfalse.de/fda <https://www.fda.gov/AboutFDA/CentersOffices/OfficeofMedicalProductsandTobacco/CDER/ucm180482.htm>

s.binfalse.de/j8u101 <http://www.oracle.com/technetwork/java/javase/8u101-relnotes-3021761.html>

s.binfalse.de/jar <http://docs.oracle.com/javase/6/docs/technotes/guides/jar/jar.html>

s.binfalse.de/jtrust-le <https://drissamri.be/blog/2017/02/22/trusting-lets-encrypt-java/>

s.binfalse.de/NCX-block <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/e/5303/5305/5319/5321/5323/5325/5327/5337/5339/5295/5345/5347/5349/5351/show/-2114380853/displayPlotFlot>

Appendix D. List of short URLs

s.binfalse.de/novak93diff http://budhat-dev.sems.uni-rostock.de/?diffModA=Novak1993_M_phase_control&diffVersA=2013-06-18&diffModB=Novak1993_M_phase_control&diffVersB=2013-11-03

s.binfalse.de/plosrepro <http://blogs.plos.org/everyone/2012/08/14/plos-one-launches-reproducibility-initiative/>

s.binfalse.de/pmr-calzone07 <http://models.cellml.org/exposure/1a3f36d015121d5596565fe7d9afb332>

s.binfalse.de/pmr-novak93 http://models.cellml.org/exposure/1e1bee6ef3243503e7e1531cfd61bb3f/novak_tyson_1993_b.cellml/view

s.binfalse.de/pmr-novak97 http://models.cellml.org/exposure/e24887f982e9246d05ba0f7152bd4aaa/novak_tyson_1997.cellml/view

s.binfalse.de/rest-curve <https://travis.cs.ox.ac.uk/FunctionalCuration/compare/e/5422/5423/5607/5608/show/-1905553820/displayPlotFlot>

s.binfalse.de/retrievalconnector <https://github.com/SemsProject/M2CAT/blob/master/src/main/java/de/unirostock/sems/M2CAT/connector/RetrievalConnector.java>

s.binfalse.de/sbmlDIFFannotator <https://github.com/binfalse/BiVeS-SBML/blob/master/src/main/java/de/unirostock/sems/bives/sbml/algorithm/SBMLDiffAnnotator.java>

s.binfalse.de/seek-demo https://demo.sysmo-db.org/models/44/compare_versions?other_version=8&version=7

s.binfalse.de/seek-docker <http://docs.seek4science.org/tech/docker/basic-container.html>

s.binfalse.de/ts140 <https://support.sas.com/techsup/technote/ts140.pdf>

s.binfalse.de/ts140-2 https://support.sas.com/techsup/technote/ts140_2.pdf

s.binfalse.de/webtools-api http://sysbioapps.dyndns.org/SED-ML_Web_Tools/Home/API

s.binfalse.de/zip <https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT>

APPENDIX E

COMODI

E.1. COMODI class hierarchy

Release 2017-10-11 of the COMODI ontology consists of 65 classes and five object properties. The object properties are listed in Table 2.1. The classes are organised into the five branches (i) Change, (ii) XmlEntity, (iii) Intention, (iv) Reason, and (v) Target (compare Figure 2.6). The namespace for all COMODI terms is <http://purl.uni-rostock.de/comodi/comodi#>. The complete class hierarchy is listed in the following. The term's name can be prepended with COMODI's namespace to get more information about it. Readers of the digital version of this thesis may also click the terms.

1. Change

- Deletion
- Insertion
- Move
 - PermutationOfEntities
- Update

2. Intention

- Correction
- Elaboration
- Expansion
 - Merge
- Simplification

Appendix E. COMODI

- Trial

3. Reason

- ChangedSpecification
- KnowledgeGain
- MismatchWithPublication
- ModelCuration
- Typo

4. Target

- ModelAnnotation
 - Date
 - CreationDate
 - ModificationDate
 - OntologyReference
 - Person
 - Contributor
 - Creator
 - TextualDescription
 - Attribution
- ModelBehaviour
- ModelDefinition
 - MathematicalModelDefinition
 - ComponentDefinition
 - EventDefinition
 - FunctionDefinition
 - KineticsDefinition
 - RuleDefinition
 - UnitDefinition
 - NetworkDefinition
 - HierarchyDefinition
 - PortDefinition
 - ReactionNetworkDefinition
 - ParticipantDefinition
 - ReactionDefinition

```

    — VariableConnectionDefinition

— ModelEncoding
    — AnnotationEncoding
        — VcardEncoding
    — IdentifierEncoding
        — MetaIdEncoding

— ModelSetup
    — ParameterSetup
    — SpeciesSetup
    — VariableSetup

5. XmlEntity
    — XmlAttribute
        — EntityIdentifier
            — ModelId
        — EntityName
            — ModelName

    — XmlNode
    — XmlText

```

E.2. COMODI's representation in Masymos

As described in Section 3.3, the COMODI ontology was integrated into the graph-based database Masymos. Thus, changes between model versions stored in Masymos can be semantically described using terms from COMODI, which makes it possible to integrate further knowledge about modifications and to implement filters for specific changes. COMODI's representation in Masymos is a graph of nodes, see Figure E.1. Every node represents a term in COMODI. The nodes are connected using an *isA* relationship according to COMODI's class hierarchy, see Appendix E.1. Nodes in Masymos, which encode for a difference between versions of a computational model, can therefore directly be linked to terms from the COMODI ontology, see Figure 3.6.

Appendix E. COMODI

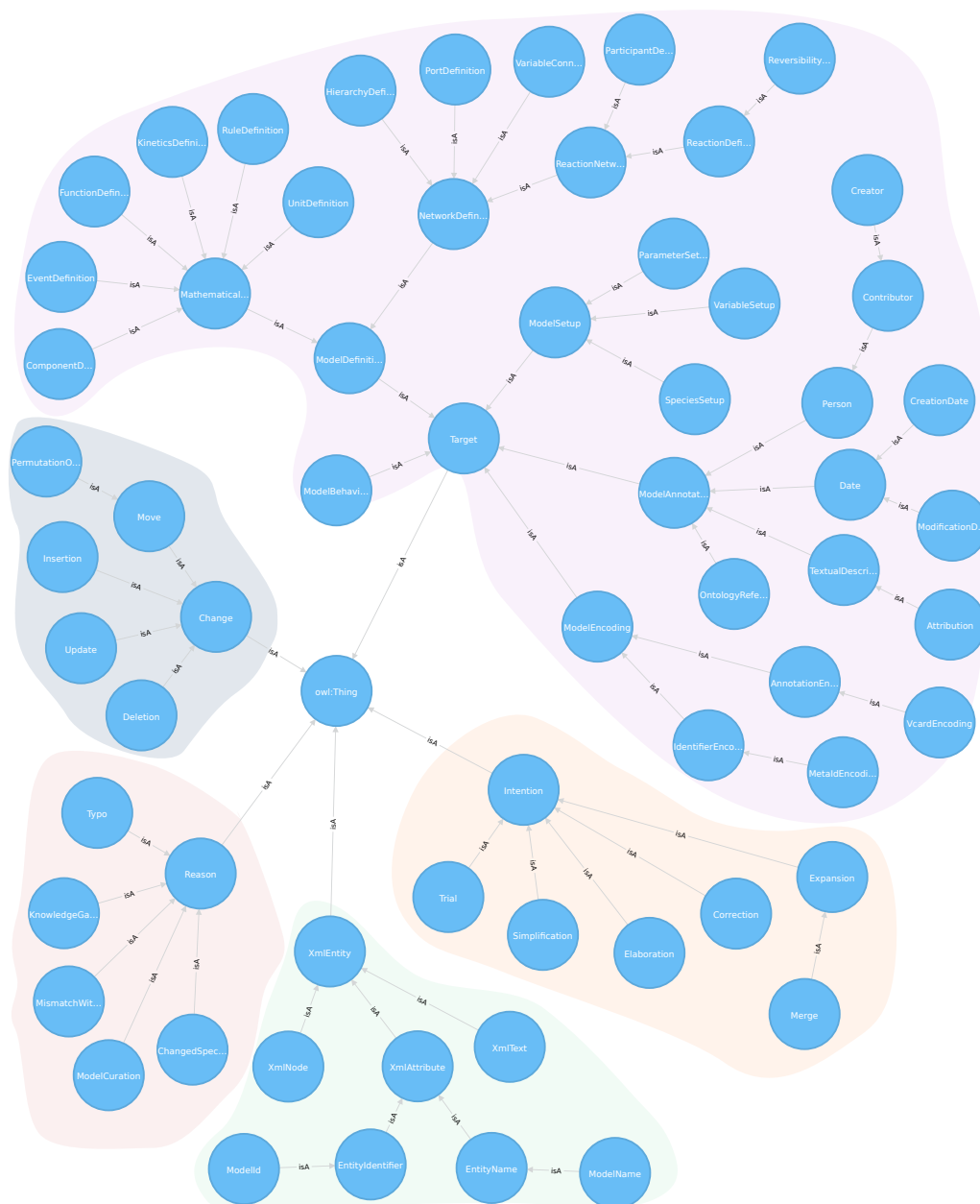


Figure E.1. COMODI's representaiton in Masymos. Terms are connected through *isA* relationships. The graph was obtained from Neo4J's web interface. I added some background colour to indicate the different branches. Compare the structure with Figure 2.6.

APPENDIX F

DEMO MODELS USED IN MASYMOS

The demonstration of the versioning concept for Masymos in Figure 3.6 was produced using two artificially created toy models in SBML. I adapted the models from [Pet16]; they are shown in Source Code F.1. The original version encodes for a reaction $A \rightarrow B$. In the modified version the reaction is updated to $A \rightarrow B + C$. In addition to the insertion of a new species, the setup of species A was changed: Originally it had an initial concentration of 100, which was updated to 120. The differences, as identified and reported by BiVeS, are shown in Source Code F.2. As explained previously, BiVeS is able to automatically annotation the differences with terms from COMODI. The annotations can, for example, be obtained using the `--separateAnnotations` flag on the command line. For the diff of the toy models BiVeS produces the annotations shown in Source Code F.3. For this example I used BiVeS version 1.11.1, which can, for example, be obtained from our web server¹.

¹bin.bio.informatik.uni-rostock.de/BiVeS, accessed 13 October 2017

Appendix F. Demo models used in Masymos

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <sbml xmlns="http://.../level2/version3"
3   ↪ level="2" version="3">
4   <model name="test_model">
5     <listOfCompartments>
6       <compartment id="default"
7         ↪ name="Default Compartment"
8         ↪ size="1" />
9     </listOfCompartments>
10    <listOfSpecies>
11      <species id="specA" name="A"
12        ↪ compartment="default"
13        ↪ initialConcentration="100" />
14      <species id="specB" name="B"
15        ↪ compartment="default"
16        ↪ initialConcentration="0" />
17    </listOfSpecies>
18    <listOfReactions>
19      <reaction id="r" name="R">
20        <listOfReactants>
21          <speciesReference
22            ↪ species="specA" />
23        </listOfReactants>
24        <listOfProducts>
25          <speciesReference
26            ↪ species="specB" />
27        </listOfProducts>
28      </reaction>
29    </listOfReactions>
30  </model>
31 </sbml>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <sbml xmlns="http://.../level2/version3"
3   ↪ level="2" version="3">
4   <model name="test_model">
5     <listOfCompartments>
6       <compartment id="default"
7         ↪ name="Default Compartment"
8         ↪ size="1" />
9     </listOfCompartments>
10    <listOfSpecies>
11      <species id="specA" name="A"
12        ↪ compartment="default"
13        ↪ initialConcentration="120" />
14      <species id="specB" name="B"
15        ↪ compartment="default"
16        ↪ initialConcentration="0" />
17      <species id="specC" name="C"
18        ↪ compartment="default"
19        ↪ initialConcentration="0" />
20    </listOfSpecies>
21    <listOfReactions>
22      <reaction id="r" name="R">
23        <listOfReactants>
24          <speciesReference
25            ↪ species="specA" />
26        </listOfReactants>
27        <listOfProducts>
28          <speciesReference
29            ↪ species="specB" />
30          <speciesReference
31            ↪ species="specC" />
32        </listOfProducts>
33      </reaction>
34    </listOfReactions>
35  </model>
36 </sbml>
```

Source Code F1. Toy models used to demonstrate the versioning concepts on a database layer.

The code shows the two models used to create Figure 3.6. The original version is shown on the left, the modified version is on the right. The changes include (i) insertion of species *c*, (ii) addition of *c* to the products of reaction *R*, and (iii) updating the initial concentration of species *A*. For a better presentation in this document I shortened the namespaces `http://www.sbml.org/sbml/level2/version3` in the root nodes to `http://.../level2/version3`.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bives type="fullDiff" id="bivesPatch">
3   <!--BiVeS compiled with: [BiVeS Framework v1.11.1] [BiVeS Core v1.8.3] [BiVeS SBML
   ↪ v1.8.2] [BiVeS CellML v1.7.3] -->
4   <update>
5     <attribute name="initialConcentration" id="1" oldValue="100" newValue="120"
   ↪   oldPath="/sbml[1]/model[1]/listOfSpecies[1]/species[1]"
   ↪   newPath="/sbml[1]/model[1]/listOfSpecies[1]/species[1]" />
6   </update>
7   <delete />
8   <insert>
9     <node id="2" newParent="/sbml[1]/model[1]/listOfSpecies[1]" newChildNo="3"
   ↪   newPath="/sbml[1]/model[1]/listOfSpecies[1]/species[3]" newTag="species" />
10    <attribute name="compartment" id="3" triggeredBy="2" newValue="default"
   ↪   newPath="/sbml[1]/model[1]/listOfSpecies[1]/species[3]" />
11    <attribute name="id" id="4" triggeredBy="2" newValue="specC"
   ↪   newPath="/sbml[1]/model[1]/listOfSpecies[1]/species[3]" />
12    <attribute name="initialConcentration" id="5" triggeredBy="2" newValue="0"
   ↪   newPath="/sbml[1]/model[1]/listOfSpecies[1]/species[3]" />
13    <attribute name="name" id="6" triggeredBy="2" newValue="C"
   ↪   newPath="/sbml[1]/model[1]/listOfSpecies[1]/species[3]" />
14    <node id="7"
   ↪   newParent="/sbml[1]/model[1]/listOfReactions[1]/reaction[1]/listOfProducts[1]"
   ↪   newChildNo="2" newPath="/sbml[1]/model[1]/listOfReactions[1]/reaction[1]/
   ↪   listOfProducts[1]/speciesReference[2]" newTag="speciesReference" />
15    <attribute name="species" id="8" triggeredBy="7" newValue="specC"
   ↪   newPath="/sbml[1]/model[1]/listOfReactions[1]/reaction[1]/listOfProducts[1]/
   ↪   speciesReference[2]" />
16  </insert>
17  <move />
18 </bives>
```

Source Code F2. Differences between the toy models reported by BiVeS. The result was obtained by executing BiVeS as `java -jar BiVeS-1.11.1-jar-with-dependencies.jar --SBML version1.xml version2.xml`.

Appendix F. Demo models used in Masymos

```
1 <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:prov="http://www.w3.org/ns/prov#"
4   xmlns:pav="http://purl.org/pav/"
5   xmlns:ore="http://www.openarchives.org/ore/terms/"
6   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
7   xmlns:comodi="http://purl.uni-rostock.de/comodi/comodi#"
8   xmlns:foaf="http://xmlns.com/foaf/0.1/">
9 <prov:Activity rdf:about="file://bives-differences.patch#createPatch">
10   <prov:generated>
11     <ore:Aggregation rdf:about="file://bives-differences.patch#bivesPatch">
12       <ore:aggregates>
13         <comodi:Insertion rdf:about="file://bives-differences.patch#8">
14           <comodi:affects rdf:resource="http://purl.uni-
15             rostock.de/comodi/comodi#ParticipantDefinition"/>
16           <comodi:appliesTo
17             rdf:resource="http://purl.uni-rostock.de/comodi/comodi#XmlAttribute"/>
18           <comodi:wasTriggeredBy>
19             <comodi:Insertion rdf:about="file://bives-differences.patch#7">
20               <comodi:affects rdf:resource="http://purl.uni-
21                 rostock.de/comodi/comodi#ParticipantDefinition"/>
22               <comodi:appliesTo
23                 rdf:resource="http://purl.uni-rostock.de/comodi/comodi#XmlNode"/>
24             </comodi:Insertion>
25           </comodi:wasTriggeredBy>
26         </comodi:Insertion>
27       </ore:aggregates>
28     <ore:aggregates rdf:resource="file://bives-differences.patch#7"/>
29     <ore:aggregates>
30       <comodi:Insertion rdf:about="file://bives-differences.patch#6">
31         <comodi:appliesTo
32           rdf:resource="http://purl.uni-rostock.de/comodi/comodi#EntityName"/>
33         <comodi:appliesTo
34           rdf:resource="http://purl.uni-rostock.de/comodi/comodi#XmlAttribute"/>
35         <comodi:wasTriggeredBy>
36           <comodi:Insertion rdf:about="file://bives-differences.patch#2">
```

```
31         <comodi:affects rdf:resource="http://purl.uni-
32             ↳ rostock.de/comodi/comodi#SpeciesSetup"/>
33         <comodi:appliesTo
34             ↳ rdf:resource="http://purl.uni-rostock.de/comodi/comodi#XmlNode"/>
35         </comodi:Insertion>
36     </comodi:wasTriggeredBy>
37 </comodi:Insertion>
38 </ore:aggregates>
39 <ore:aggregates>
40     <comodi:Insertion rdf:about="file://bives-differences.patch#5">
41         <comodi:affects
42             ↳ rdf:resource="http://purl.uni-rostock.de/comodi/comodi#SpeciesSetup"/>
43         <comodi:appliesTo
44             ↳ rdf:resource="http://purl.uni-rostock.de/comodi/comodi#XmlAttribute"/>
45         <comodi:wasTriggeredBy rdf:resource="file://bives-differences.patch#2"/>
46     </comodi:Insertion>
47 </ore:aggregates>
48 <ore:aggregates>
49     <comodi:Insertion rdf:about="file://bives-differences.patch#4">
50         <comodi:appliesTo rdf:resource="http://purl.uni-
51             ↳ rostock.de/comodi/comodi#EntityIdentifier"/>
52         <comodi:appliesTo
53             ↳ rdf:resource="http://purl.uni-rostock.de/comodi/comodi#XmlAttribute"/>
54         <comodi:wasTriggeredBy rdf:resource="file://bives-differences.patch#2"/>
55     </comodi:Insertion>
56 </ore:aggregates>
57 <ore:aggregates>
58     <comodi:Update rdf:about="file://bives-differences.patch#1">
59         <comodi:affects
60             ↳ rdf:resource="http://purl.uni-rostock.de/comodi/comodi#SpeciesSetup"/>
```

Appendix F. Demo models used in Masymos

```
60     <comodi:appliesTo
        ↪   rdf:resource="http://purl.uni-rostock.de/comodi/comodi#XmlAttribute"/>
61     </comodi:Update>
62   </ore:aggregates>
63   <ore:aggregates rdf:resource="file://bives-differences.patch#2"/>
64   <rdf:type rdf:resource="http://www.w3.org/ns/prov#Entity"/>
65 </ore:Aggregation>
66 </prov:generated>
67 <prov:wasAssociatedWith>
68   <prov:SoftwareAgent rdf:about="file://bives-differences.patch#bives">
69     <pav:version>BiVeS compiled with: [BiVeS Framework v1.11.1] [BiVeS Core v1.8.3]
        ↪   [BiVeS SBML v1.8.2] [BiVeS CellML v1.7.3] </pav:version>
70     <rdfs:label>BiVeS</rdfs:label>
71   </prov:SoftwareAgent>
72 </prov:wasAssociatedWith>
73 </prov:Activity>
74 </rdf:RDF>
```

Source Code F3. Annotations of the differences between the toy models reported by BiVeS. The result was obtained by executing BiVeS as `java -jar BiVeS-1.11.1-jar-with-dependencies.jar --SBML --separateAnnotations version1.xml version2.xml`.

APPENDIX G

SOFTWARE DEVELOPED FOR THIS THESIS

During my time as a PhD candidate, I always tried to materialise my ideas into code. Thus, a number of tools emerged from this thesis. While they are all relevant for my thesis, not every tool managed to get mentioned in this document. Some of the tools are the result of successful collaborations. The following sections will briefly present some of the tools, summarised in Table G.1. All tools are free software: Everyone is allowed to run, study, modify, and redistribute them¹. The source code can typically be found at GitHub² and compiled binaries are available from bin.bio.informatik.uni-rostock.de. Tools developed in Java typically implement support for Maven, and thus can easily be integrated in other software projects. They are either packaged under the group-id `de.uni-rostock.sbi`³ or `de.uni-rostock.sems`⁴. In the following I will describe the tools from a rather technical perspective.

¹fsfe.org/freesoftware/basics/4freedoms.en.html, accessed 8 February 2018

²github.com/binfalse, accessed 8 February 2018

³Available from search.maven.org/#search%7Cga%7C1%7Cg%3A%22de.uni-rostock.sbi%22, accessed 8 February 2018

⁴Available from our third-party Maven repository mvn.bio.informatik.uni-rostock.de, accessed 8 February 2018

G.1. BiVeS

BiVeS implements my algorithm to detect and communicate differences in computational models, see Section 2.2. It is written in Java and, thus, operating system independent. BiVeS consists of a number of modules, as shown in Figure 2.3. All modules are developed at GitHub; the repositories can be found at:

BiVeS-Core github.com/binfalse/BiVeS-Core

BiVeS-SBML github.com/binfalse/BiVeS-SBML

BiVeS-CellML github.com/binfalse/BiVeS-CellML

BiVeS github.com/binfalse/BiVeS

BiVeS-WebApp github.com/binfalse/BiVeS-WebApp

BiVeS-WebApp-Client github.com/binfalse/BiVeS-WebApp-Client

BiVeS-Core provides the infrastructure for the BiVeS framework, including API methods, general mapping strategies and means to communicate identified differences. BiVeS-SBML and BiVeS-CellML make BiVeS understand the domain characteristics of models encoded in SBML and CellML, respectively. The BiVeS module implements a command line interface and bundles the framework. Unnecessary modules can be discarded (e.g. the developers of a repository for models in CellML format do not need to compile the SBML module into BiVeS). In addition, the modular design of BiVeS is ready to support further standard formats.

BiVeS-WebApp implements a web interface on top of BiVeS. Models can be submitted through HTTP POST requests and the comparison will be done on the server side. Thus, BiVeS can be used for virtually any tool and PC that has an internet connection. The BiVeS-WebApp is also available as a Docker image from the Docker Hub as [binfalse/bives-webapp](https://hub.docker.com/r/binfalse/bives-webapp)⁵, which makes it very easy to deploy a new instance of BiVeS. A public instance of the application is available at bives.bio.informatik.uni-rostock.de. The BiVeS-WebApp-Client library supports developers in integrating BiVeS into other software projects.

G.2. BiVeS-StatsGenerator

The BiVeS-StatsGenerator implements the pipeline introduced in Section 3.4. It uses the ModelCrawler⁶ to retrieve all models and their versions from BioModels Database and the Physiome Model Repository. The files are restructured to facilitate subsequent processing. Model versions are then compared using Unix' diff and BiVeS and the results are exported into *data tables*. These tables collect figures on model properties and information on the differences, cf. Section 3.4.1. The results are, for example, used by the MoSt tool to allow for interactive access to the evolution of models, see Section 3.4.2.

⁵hub.docker.com/r/binfalse/bives-webapp, accessed 8 February 2018

⁶github.com/FreakyBytes/ModelCrawler, accessed 20 October 2017

G.3. BudHat

BudHat was my first prototypic implementation of a versioning system to demonstrate the impact of BiVeS. BudHat is a web-based interface developed in Java and uses BiVeS through its Java API. The first version used to use Cytoscape Web⁷ and can be found at budhat.bio.informatik.uni-rostock.de. In the second version I implemented support for Cytoscape.js⁸; it can be found at budhat-dev.bio.informatik.uni-rostock.de. BudHat always recomputes the differences online and shows the results reported by BiVeS in different formats. However, as there are already many other real-life showcases demonstrating the power of BiVeS (compare Section 3), BudHat is discontinued and only survives for the references.

G.4. CaRo

I compared the approaches behind COMBINE archives and Research Objects and found both formats quite similar. Sure, they focus and emphasise different aspects and use other formats to store metadata. However, both approaches represent containers consisting of (i) a manifest, (ii) metadata, and (iii) files that are related to a research result and, ultimately, cut the ground from each other. Therefore, I developed the CaRo library, which is able to convert a COMBINE archive into a Research Object and vice versa. CaRo is written in Java and available from GitHub at github.com/binfalse/CaRo. It is in a pre-release stage, because there are still subtle but important aspects that are not perfectly translatable into each other. For example, COMBINE archives allow for master files – files that are supposed to be main entries of that container. There is no such concept for Research Objects. Thus, a converted Research Object may not necessarily represent the same aspects of a research result. We are, however, working on harmonizing the standards.

In Addition, I developed CaRo Web, which builds on top of CaRo and provides a web interface to convert between COMBINE archives and Research Objects. There are two endpoints (i) `/caro` to convert a COMBINE archive into a Research Object and (ii) `/roca` to convert a Research Object into a COMBINE archive. The development also takes place at GitHub: github.com/binfalse/CaRoWeb. CaRo Web is also available as a Docker image from the Docker Hub as `binfalse/caroweb`⁹. A public instance of CaRo Web is available at caro.bio.informatik.uni-rostock.de.

⁷cytoscapeweb.cytoscape.org, accessed 20 October 2017

⁸js.cytoscape.org, accessed 20 October 2017

⁹hub.docker.com/r/binfalse/caroweb, accessed 8 February 2018

G.5. CombineArchive library

The CombineArchive library is a Java library to read, write, create, and manipulate COMBINE archives, see Section 4.4.1. It is fully compliant with the latest COMBINE archive specification identifiers.org/combine.specifications/omex. In addition to the proposed metadata format, the library supports any other XML-based annotations of files included in an archive. It is developed at GitHub: github.com/SemsProject/CombineArchive. An example on how to create and read a COMBINE archive is shipped with the code¹⁰.

G.6. CombineArchiveWeb application

The CombineArchiveWeb application builds on the CombineArchive library and provides a web platform to explore and share COMBINE archives, see Section 4.4.2. COMBINE archives can be developed in so-called workspaces. A workspace may contain multiple COMBINE archives and a user may have multiple workspaces. Individual workspaces can be shared with others to develop COMBINE archives collaboratively.

The CombineArchiveWeb application was designed with portability and an easy installation in mind. Therefore, it does not require a database back-end and there is no complicated mechanism to authenticate users. The storage concept rests on a sophisticated filesystem structure: Workspaces are represented by directories, which may contain COMBINE archives. For both the workspace identifiers and the COMBINE archive identifiers the CombineArchiveWeb application uses Universally Unique Identifiers¹¹ (UUID). To manage workspaces a Java Properties¹² file is used, which, e.g., contains mappings between names and identifiers. The authentication boils down to the knowledge of workspace identifiers: If a user knows the UUID associated to a workspace he has access to that workspace. The workspaces of a user are stored in the user's cookies. Thus, workspaces are browser- and profile-dependent, but can easily be shared through links of the format `https://DOMAIN/rest/share/WORKSPACE_UUID`. This endpoint then just extends the cookie value of the user with the corresponding workspace UUID.

Space constraints can be implemented using quotas in the configuration of the CombineArchiveWeb application. This way it is, for example, possible to limit the maximum size of a workspace or an archive. The application also tracks last-access times of workspaces, which makes it possible to remove orphaned workspaces.

A public instance of the CombineArchiveWeb application is available at cat.bio.informatik.uni-rostock.de. As the application is also available as a Docker image from the Docker Hub as `binfalse/webcat`¹³, it is fairly easy to install and maintain an individual instance.

¹⁰s.binfalse.de/ca-example, accessed 23 October 2017

¹¹en.wikipedia.org/wiki/Universally_unique_identifier, accessed 8 February 2018

¹²docs.oracle.com/javase/9/docs/api/java/util/Properties.html, accessed 8 February 2018

¹³hub.docker.com/r/binfalse/webcat, accessed 8 February 2018

G.7. CombineExt

The CombineExt library is a collection of extensions, which eases working with COMBINE-based standards in Java tools. The library is, for example, able to recognise the format of a file encoded in a COMBINE standard (such as CellML, SBML, or SBGN-ML) and to determine the corresponding Identifiers.org URI corresponding to the specific standard format. For example, given a model document in SBML L1V2 format, the CombineExt library recognises the file type and returns `http://identifiers.org/combine.specifications/sbml.level-1.version-2` as the format URI. It thereby evaluates the contents of a file, its media type, or its extension to guess the file's format. In addition, the CombineExt library provides icons for COMBINE standards. Given a COMBINE document, it returns an `InputStream` to read the icon image from. Available icons are, for example, shown in the library's documentation¹⁴.

The library was initially developed for the CombineArchiveWeb application, which needs to recognise formats and displays icons for different file types. The structure of CombineExt makes it easy to implement support for further formats. To demonstrate how the library can be extended I developed CombineExt-PharmML¹⁵, which extends CombineExt with support for PharmML.

G.8. Functional Curation WebLab

The Functional Curation WebLab is a web-based repository of simulation studies, which I developed during my internship in the Computational Biology Group¹⁶ at the University of Oxford¹⁷. The idea behind the WebLab explicitly distinguishes between models and protocols, which can be linked to each other dynamically. The links in these so-called virtual experiments are realised through annotations with terms from ontologies. The concept is described in more detail in Section 4.6.

The WebLab platform is based on Java and uses a MySQL database to store models, protocols, simulation results, and user data. It provides basic password-based authentication for users, who can upload new models or protocols, and (re)submit virtual experiments. The WebLab integrates the CombineArchive library to package virtual experiments as COMBINE archives, which at least consist of a model and a protocol. These experiments can then be sent to a compute node, which is able to understand the simulation job encoded in the COMBINE archive and runs the simulation using the Chaste software library [Mir+13]. Once Chaste has finished its calculations, the simulation results are again packaged into a COMBINE archive and sent back to the WebLab. The Functional Curation WebLab makes it easy to compare the results of

¹⁴semsproject.github.io/CombineExt/CombineIconizer, accessed 8 February 2018

¹⁵semsproject.github.io/CombineExt-PharmML, accessed 8 February 2018

¹⁶www.cs.ox.ac.uk/research/compbio/index.html, accessed 27 February 2018

¹⁷www.ox.ac.uk, accessed 27 February 2018

Appendix G. Software developed for this thesis

different experiments, see for instance Figures 4.10 and 4.11. To visualise the results the WebLab integrated three different JavaScript libraries: Flot¹⁸, HighCharts¹⁹, and D3²⁰. Simulation results can be downloaded file-wise, or packaged as COMBINE archive. A sophisticated URL structure realises shareable permanent links using identifiers from the database back-end. For example, the link `.../compare/e/5651/5652/5660/show/1848458697/displayPlotHC`²¹ compares the experiments *Carro 2011 Endo & Steady state 2Hz pacing* (identifier 5651), *Carro 2011 Epi & Steady state 2Hz pacing* (identifier 5652), and *Fink 2008 & Steady state 2Hz pacing* (identifier 5660). More specifically, it shows the comparison of the file *outputs_Final_pace_voltage_gnuplot_data.csv* (identifier 1848458697) using the HighCharts library (`displayPlotHC`).

The whole WebLab was designed with version control in mind. That is, all entities may exist in multiple versions. New versions of models and protocols can be uploaded without overwriting or invalidating old ones. Similarly, new versions of simulation results can be generated by resubmitting an experiment. All (public) versions of an entity are accessible and comparable through the WebLab's portal²². The files in different version can be compared using Unix' diff or BiVeS, if applicable. For example, Figure 3.3 shows the comparison of two model versions at the Functional Curation WebLab.

G.9. jComodi

The COMODI ontology empowers users and software to describe changes in a model on the semantic level, see Section 2.4. In addition, I devised the jComodi Java library to assist developers in implementing support for COMODI in Java-based software tools. Using jComodi, it is easy to read and export changes between (versions of) models. jComodi also integrates Apache's Jena²³ and is, thus, able to handle considerably complex annotations. A *ChangeFactory*²⁴ can be used to generate annotations. Annotations can be serialised as either RDF/XML or TURTLE. jComodi is already integrated in the BiVeS framework. BiVeS provides an abstract class *DefaultDiffAnnotator*²⁵ and two implementations of it (*SBMLDiffAnnotator*²⁶ and *CellMLDiffAnnotator*²⁷), which use jComodi to automatically annotate identified differences between versions of computational models.

¹⁸www.flotcharts.org, accessed 27 February 2018

¹⁹www.highcharts.com, accessed 27 February 2018

²⁰d3js.org, accessed 27 February 2018

²¹travis.cs.ox.ac.uk/FunctionalCuration/compare/e/5651/5652/5660/show/1848458697/displayPlotHC, accessed 27 February 2018

²²see for example the different versions of the Decker 2009 model s.binfalse.de/decker-versions, accessed 27 February 2018

²³jena.apache.org, accessed 28 February 2018

²⁴s.binfalse.de/changefactory, accessed 28 February 2018

²⁵s.binfalse.de/defaultdiffannotator, accessed 28 February 2018

²⁶s.binfalse.de/sbmldiffannotator, accessed 28 February 2018

²⁷s.binfalse.de/cellmldiffannotator, accessed 28 February 2018

G.10. M2CAT

M2CAT is a web-based platform to extract reproducible simulation studies using the CombineArchive Toolkit. It enables researchers to retrieve models or simulation studies from Masymos and directly generates COMBINE archives from all relevant files. It was described in detail in Section 4.5. I initially built a prototype and published it at the BTW conference in 2015 [SW15]. The original code is available at github.com/binfalse/Masymos2CAT. The platform was, however, fully refurbished by a group of students, who I supervised.

M2CAT's web interface provides a search field to query a Neo4J database (Masymos) for simulation studies. The link to Masymos is established through the Morre client library²⁸. Using Morre, M2CAT is for example able to discover simulation studies based on model annotations or authors. Matching studies can be enriched with additional files and metadata. So-called connectors provide connections to third party repositories and web interfaces. A connector needs to implement a `RetrievalConnector` interface²⁹. Various connectors already exist, for example, obtain curation results from BioModels Database, additional files from the Physiome Model Repository, and metadata from Pubmed. The project was designed to also support other third-party connectors.

M2CAT integrates the CombineArchive library. Thus, simulation studies can be downloaded as a COMBINE archive or exported to the CombineArchiveWeb application, cf. Figure 4.7. The configuration of M2CAT is possible through context variables of the Java webserver³⁰. The compiled application is also available as a Docker image from the Docker Hub as `binfalse/m2cat`³¹.

G.11. MoSt

MoSt is a web platform to interactively explore the evolution of computational models, see Section 3.4. It is primarily written in PHP and JavaScript, and does not require a database. Instead, MoSt uses the resulting data tables of the Statistics Generator to visualise the history of models, cf. Section 3.4.1. MoSt implements a sophisticated filter mechanism. Models can, for example, be filtered by type, date, and identifiers³². The filters are reflected in the browser's address bar. For example, `http://most.sems.uni-rostock.de/#t:s,d1:2004-12-31,d2:2017-12-31,m:BIOMD0000000426&BIOMD0000000312` filters for SBML models (`t:s`), which were created between 31 December 2004 (`d1:2004-12-31`) and 31 December 2017 (`d2:2017-12-31`), and are either identified by `BIOMD0000000426` or `BIOMD0000000312`. Thus, it is also possible to link to the evolution of all curated models which emerged from BioModels Database (`m:BIOMD0`), or to the full history of a specific model (e.g. `m:BIOMD0000000312`).

²⁸github.com/binfalse/morre-client, accessed 28 February 2018

²⁹s.binfalse.de/retrievalconnector, accessed 28 February 2018

³⁰semsproject.github.io/M2CAT/config, accessed 28 February 2018

³¹hub.docker.com/r/binfalse/m2cat/, accessed 28 February 2018

³²github.com/binfalse/MOST#filter-the-data, accessed 28 February 2018

Appendix G. Software developed for this thesis

Tool	License	Description
BiVeS	Apache 2.0	implements difference detection for computational models, see Chapter 3, BiVeS actually consists of multiple modules (github.com/binfalse/BiVeS)
BiVeS-StatsGenerator	Apache 2.0	implements the pipeline to analyse models in open repositories, introduced in Figure 3.7 (github.com/binfalse/BiVeS-StatsGenerator)
BudHat	Apache 2.0	prototypic implementation of a web-based versioning system using the BiVeS framework (budhat.sems.uni-rostock.de)
CaRo	LGPL v3.0	converter between COMBINE archives and Research Objects (github.com/binfalse/CaRo)
CombineArchive library	BSD 3-clause	Java library to read, write, create, and manipulate COMBINE archives, see Section 4.4.1 (github.com/binfalse/combinearchive)
CombineArchive Web	GPL v3.0	web interface to create/modify/share CombineArchives, see Section 4.4.2 (cat.bio.informatik.uni-rostock.de)
CombineExt	LGPL v3.0	library that supports developers in working with COMBINE standards (github.com/binfalse/CombineExt)
Functional Curation WebLab	BSD 3-Clause	web portal for functional curation of computational models, see Section 4.6 (travis.cs.ox.ac.uk/FunctionalCuration)
jComodi	Apache 2.0	Java client supporting developers in using the COMODI ontology, see Section 2.4 (github.com/binfalse/jCOMODI)
M2CAT	GPL v3.0	web-based tool to find and export reproducible research results, see Section 4.5 (github.com/binfalse/M2CATv2)
MoSt	Apache 2.0	web portal for interactive access to the evolution of models, see Section 3.4.2 (github.com/SemsProject/MoSt)

Table G.1. A collection of tools and libraries developed in the scope of this thesis.

MoSt’s visualisations are generated using D3³³. In addition, MoSt includes the BiVeS web application to recompute deltas online. The differences are then shown in form of (i) the human-readable report, (ii) the highlighted reaction network rendered using DiViI, (iii) the actual delta encoded in XML format, and (iv) the annotations highlighted graphically in the network of COMODI terms, cf. Figure 3.11.

³³d3js.org, accessed 27 February 2018

APPENDIX H

CURRICULUM VITAE

Martin Scharm

Date of Birth 27 October 1985

Place of birth Güstrow

Nationality German



Academic Background

- 2005 Abitur at the John-Brinckmann-Gymnasium in Güstrow, Germany
- 2011 Research Internship at the University of Reims Champagne-Ardenne in Reims, France
- 2012 Diploma in Bioinformatics at the Martin-Luther-University in Halle-Wittenberg, Germany
- since 2012 PhD candidate at the Department of Systems Biology and Bioinformatics of the University of Rostock in Rostock, Germany
- 2013 Research Internship at the University of Oxford in Oxford, United Kingdom
- 2015 Research Internship at the University of Manchester in Manchester, United Kingdom

APPENDIX I

ORIGINAL PUBLICATIONS

I.1. Papers in refereed journals

1. **M Scharm**, T Gebhardt, V Touré, A Bagnacani, A Salehzadeh-Yazdi, O Wolkenhauer, D Waltemath: Evolution of computational models in BioModels Database and the Physiome Model Repository. *BMC Systems Biology* (2018) 12:53.

I designed the study together with Dagmar Waltemath. I developed the tools and the pipeline used to analyse the models and their versions. Tom Gebhardt developed the MoSt web portal under my supervision. Vasundra Touré and I created the R-based figures. Vasundra Touré and Dagmar Waltemath developed the example. Together with Vasundra Touré, Tom Gebhardt, Andrea Bagnacani, and Olaf Wolkenhauer, I created the example figure. I wrote the initial version(s) of the manuscript.

2. FT Bergmann, D Nickerson, D Waltemath, **M Scharm**: SED-ML web tools: Generate, modify and export standard-compliant simulation studies. *Bioinformatics* (2017) 33:8.

The SED-ML Web Tools were initially developed by Frank T. Bergmann. Together we improved the tools and their API to allow for an easy integration in other software projects. I created the figure and wrote the initial version of the manuscript.

Appendix I. Original publications

3. D Waltemath, JR Karr, FT Bergmann, V Chelliah, M Hucka, **M Scharm**, et al.: Toward community standards and software for whole-cell modeling. *IEEE Transactions on Biomedical Engineering* (2016) 63:10.

This paper is the result of a summer school on whole-cell modelling and contains contributions of more than 50 authors. I participated in the hackathon and contributed to the paper.

4. **M Scharm**, D Waltemath: A fully featured COMBINE archive of a simulation study on syncytial mitotic cycles in *Drosophila* embryos [version 1; referees: 1 approved, 2 approved with reservations]. *F1000Research* (2016) 5:2421.

I developed the fully featured COMBINE archive. During the development I tracked provenance in great detail and wrote extensive documentation. Based on that data, Dagmar Waltemath and I crafted this paper.

5. **M Scharm**, D Waltemath, P Mendes, O Wolkenhauer: COMODI: An ontology to characterise differences in versions of computational models in biology. *Journal of Biomedical Semantics* (2016) 7:46.

I designed the study and drafted the manuscript together with Dagmar Waltemath. I developed the software and created the website. I built the initial version of the ontology together with Dagmar Waltemath. Pedro Mendes and Olaf Wolkenhauer critically revised the ontology and the manuscript.

6. J Cooper, **M Scharm**, GR Mirams: The Cardiac Electrophysiology Web Lab. *Biophysical Journal* (2016) 110:2.

The idea of virtual experiments was initially proposed in [CSM16]. During my internship at the Computational Biology Group in Oxford I designed and developed the Cardiac Electrophysiology Web Lab, based on the simulation software developed by Jonathan Cooper. Jonathan Cooper and Gary R. Mirams analysed the results. I contributed to the paper.

7. **M Scharm**, O Wolkenhauer, D Waltemath: An algorithm to detect and communicate the differences in computational models describing biological systems. *Bioinformatics* (2015) 32:4.

I developed the algorithm and implemented it in BiVeS/BudHat. I wrote the manuscript and created the figures. I crafted the example together with Dagmar Waltemath.

8. FT Bergmann, R Adams, S Moodie, J Cooper, M Glont, **M Scharm**, et al.: COMBINE archive and OMEX format: One file to share all information to reproduce a modeling project. *BMC Bioinformatics* (2014) 15:369.

The development of the COMBINE archive format was a joint effort from within the COMBINE community. Frank T. Bergmann, Richard Adams, and Nicolas Le Novère wrote the initial version of the COMBINE archive format. I critically revised the manuscript and contributed an example to the final paper.

9. D Waltemath, R Henkel, R Hälke, **M Scharm**, O Wolkenhauer: Improving the reuse of computational models through version control. *Bioinformatics* (2013) 29:6.

Dagmar Waltemath, Ron Henkel, and Olaf Wolkenhauer identified the requirements for model version control. Dagmar Waltemath and Ron Henkel developed the concepts. I (re-)implemented the software. I created the figures and critically revised the manuscript.

I.2. Publications in books and proceedings

1. **M Scharm**, NJ Stanford, PD Dobson, M Golebiewski, M Hucka, VB Kothamachu, D Nickerson, S Owen, J Pahle, U Wittig, D Waltemath, C Goble, P Mendes, J Snoep: *Data management in Computational Systems Biology: Exploring standards, tools, databases and packaging best practices*. In *Yeast Systems Biology. Methods and Protocols*; Book chapter submitted to *Springer-Nature*.
2. **M Scharm**, D Waltemath: *Extracting reproducible simulation studies from model repositories using the CombineArchive Toolkit*. In *GI-Edition Lecture Notes in Informatics (LNI)*; Workshop on Data Management for Life Sciences (DMforLS 2015) @ Datenbanksysteme für Business, Technologie und Web (BTW, 2015); Hamburg, GER.
3. **M Scharm**, F Wendland, M Peters, M Wolfien, T Theile, D Waltemath: *The CombineArchive-Web application – A web based tool to handle files associated with modelling results*. In *Proceedings of the SWAT4LS 2014*; Semantic Web Applications and Tools for Life Sciences (SWAT4LS, 2014); Berlin, GER.
4. **M Scharm**, D Waltemath, O Wolkenhauer: *Identifying, Interpreting, and Communicating Changes in XML-encoded Models of Biological Systems*. In *Proceedings of the 10th International Conference on Data Integration in the Life Sciences (DILS)*; Data Integration in the Life Sciences (2014); Lisbon, PT.

I.3. Selected talks

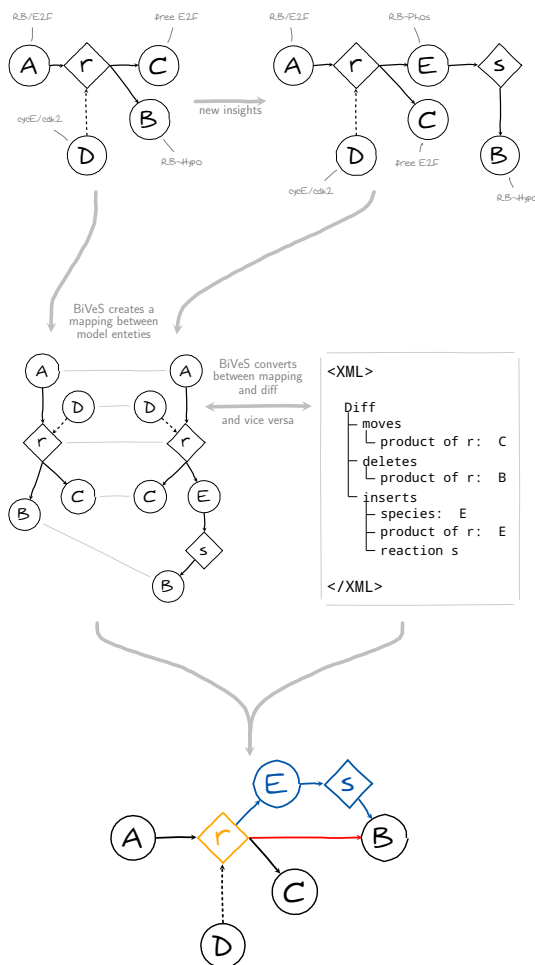
1. **M Scharm**, D Waltemath: *Model Management in Systems Biology: Challenges – Approaches – Solutions*. (Webinar) FAIRDOM webinar series (2016).
2. M Wolfien, A Bagnacani, T Gebhardt, **M Scharm**: *A Modeler's Tale*. (Invited Talk) 10th International CellML Workshop (2016); Auckland, New Zealand; Movie available from Figshare as [10.6084/m9.figshare.3423371.v1](https://figshare.com/figures/data/3423371/v1).
3. **M Scharm**, D Waltemath: *Characterising differences in model versions*. (Conference Talk) e:Bio Statusseminar (2015); Essen, GER.
4. **M Scharm**: *Sharing Reproducible Research Results*. (Invited Talk) Manchester University (2015); Manchester, UK.
5. **M Scharm**: *M2CAT: Extracting reproducible simulation studies from model repositories using the CombineArchive Toolkit*. (Conference Talk) 9th international CellML Workshop (2015); Waiheke Island, Auckland, NZ.
6. **M Scharm**, D Waltemath: *Extracting reproducible simulation studies from model repositories using the CombineArchive Toolkit*. (Conference Talk) Workshop on Data Management for Life Sciences (DMforLS 2015) @ Datenbanksysteme für Business, Technologie und Web (BTW, 2015); Hamburg, GER.
7. **M Scharm**: *The CellML models' walk through the repository: A retrospective study*. (Conference Talk) 8th international CellML Workshop (2014); Waiheke Island, Auckland, NZ.
8. **M Scharm**: *Improving the Management of Computational Models*. (Invited Talk) European Bioinformatics Institute (EBI, 2013); Cambridge, UK.
9. **M Scharm**: *BiVeS & BudHat: Difference Detection for Computational Models*. (Conference Talk) COMBINE (2013); Paris, Fr.
10. **M Scharm**: *Model version control for computational biology*. (Invited Talk) IPK (Leibniz Institute, 2013); Gatersleben, GER.
11. **M Scharm**: *BudHat: Version Control for Computational Models*. (Software Demo) International Symposium on Integrative Bioinformatics (IB, 2013); Gatersleben, GER.

I.4. Selected posters

1. **M Scharm**, O Wolkenhauer, D Waltemath: Disentangling the Evolution of Computational Models. International Symposium on Integrative Bioinformatics. IPK-Gatersleben, Germany, March 2013
2. **M Scharm**, F Wendland, M Peters, M Wolfien, T Theile, D Waltemath: The CombineArchive Toolkit – facilitating the transfer of research results. Semantic Web Applications and Tools for Life Sciences (SWAT4LS). Berlin, Germany, December 2014
3. **M Scharm**, D Waltemath, O Wolkenhauer: Identifying, Interpreting, and Communicating Changes in XML-encoded Models of Biological Systems. 10th International Conference on Data Integration in the Life Sciences (DILS). Lisbon, PT, July 2014
4. GR Mirams, **M Scharm**, J Cooper: A Web Lab for Cardiac Electrophysiology. Gordon Research Conference on Cardiac Arrhythmia Mechanisms. Lucca (Barga), Italy, March 2015

The posters are attached on the following pages.

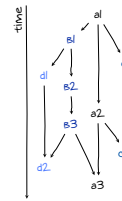
Disentangling the Evolution of Computational Models



Problem Statement

- Models evolve over time.
- New versions of a model, describing the same biological process, emerge before and after publication.
- Reconstructing the changes in a model and understanding what has been done to it pose two problems to modelers reusing existing code.

Version control helps to gain insights in the process of modelling and increases the confidence in computational models [3].



BiVeS

Biochemical Model Version Control System [2]

- Java library to map hierarchically structured content
- compares models encoded in standardized formats (currently: SBML [4] and XSBML [5])
- matches unchanged or moved entities in model documents
- identifies inserts and deletes
- constructs a diff (in XML format)

The diff produced by BiVeS can be used to grasp the changes which occurred between two versions of a model.

BudHat

BudHat is a web interface to visualize the differences identified by BiVeS.

- calls BiVeS to construct the diff
- displays the result in various formats
 - the XML diff
 - a reaction network highlighting the changes using Cytoscape [6]
 - a human readable report
- is bound to a database backend holding the models

BudHat can be integrated in existing repositories.

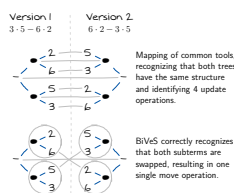
Why not simply use...

...your eyes? XML is not easy to read and even harder to compare. Moreover, models are far too complex to waste your precious time!

...unix' diff? The unix' diff tool is line-based and optimized to compare flat files. It is not able to handle hierarchical structures.

...SVN? SVN, like other common version control systems (VCSs), uses unix' diff tool to identify differences. Therefore, these VCSs are inappropriate for model version control.

...existing XML-diff detection tools? Their diffs are neither minimal nor very meaningful, see example on the right.



References

- [1] <http://sems.uni-rostock.de/budhat/>
- [2] <http://sems.uni-rostock.de/bives/>
- [3] Waltemath et al.: Improving the reuse of computational models through version control Bioinformatics, 2013
- [4] Hucka et al.: The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models Bioinformatics, 2003
- [5] Lloyd et al.: CellML: its future, present and past Progress in Biophysics and Molecular Biology, 2004
- [6] Lopes et al.: Cytoscape Web: an interactive web-based network browser Bioinformatics, 2010



Martin Scharm
University of Rostock
sems.uni-rostock.de/scharm
martin.scharm@uni-rostock.de



Olaf Wolkenhauer
University of Rostock
sems.uni-rostock.de/wolkenhauer
olaf.wolkenhauer@uni-rostock.de



Dagmar Waltemath
University of Rostock
sems.uni-rostock.de/waltemath
dagmar.waltemath@uni-rostock.de

CombineArchiveToolkit

facilitating the transfer of research results

Universität
Rostock



Traditio et Innovatio



SYSTEMS BIOLOGY
BIOINFORMATICS
ROSTOCK

SEMS
simulation experiment management system

The COMBINE Idea

The 'COmputational Modeling in Biology' Network (COMBINE) is an initiative to coordinate the development of the various community standards and formats for computational models (BioPax, SBGN, SBML, SED-ML, etc.) [1]. One of the major goals of COMBINE is to improve the interoperability of these standards, and to support fledgling efforts aimed at filling gaps or new needs.

The steadily increasing size and complexity of models and derived data poses the challenge of sharing reproducible results. Today, these results typically consist of multiple model files, simulation descriptions, publications, and meta data. The question how to provide all relevant files and modelling results, in a reliable and reproducible manner, remains.

In 2011 the COMBINE community [2] proposed the COMBINE archive format [3] which is a container that bundles all files related to a project into a single file. Typically, it comprises the model files needed to run a particular set of experiments. In addition, it contains all associated files that are needed to reproduce the experiments such as simulation experiment descriptions (SED-ML), semantic annotations, or graphical representations in SBGN-ML. All files can be equipped with meta-information such as people attributions and details about the files inside the archive. Generally, a COMBINE archive is encoded using the Open Modelling EXchange format (OMEX).

What's the gap?

Manually handling COMBINE archives is tedious and error prone. Consequently, computational support is needed to undertake this task. Only then, it will become possible to exchange COMBINE archives seamlessly between different applications and repositories. Such a tool is constrained to provide mechanisms to create, explore and modify files and meta information in a COMBINE archive.

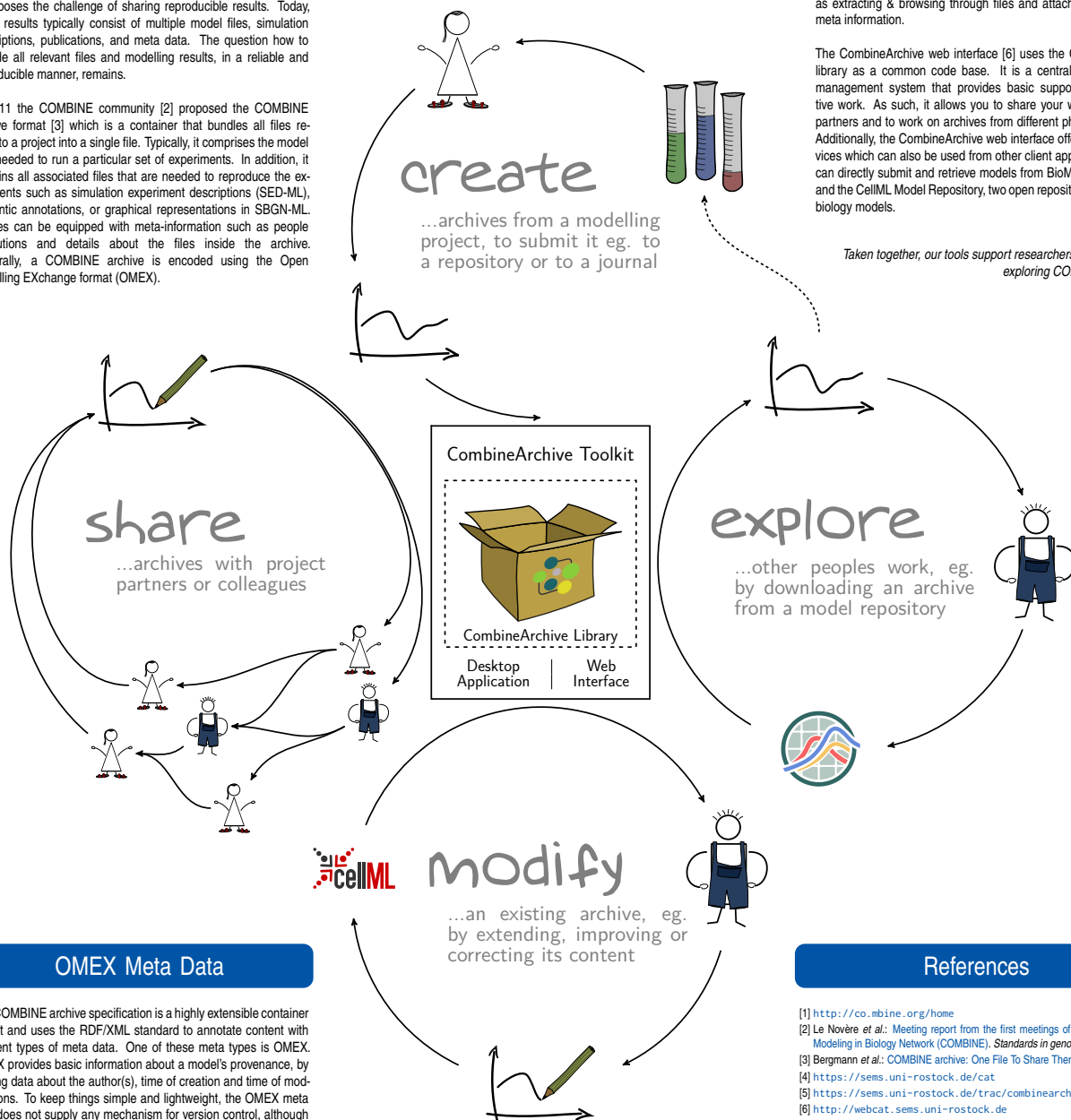
Our Approach

To provide the needed computational support, we developed the CombineArchiveToolkit [4]. It consists of a core library, a desktop application, and a web based interface. The CombineArchiveLibrary [5] was implemented using latest Java technologies. It offers all necessary methods to handle COMBINE archives, such as extracting & browsing through files and attaching & retrieving meta information.



The CombineArchive web interface [6] uses the CombineArchive library as a common code base. It is a centralised cloud data management system that provides basic support for collaborative work. As such, it allows you to share your workspaces with partners and to work on archives from different physical locations. Additionally, the CombineArchive web interface offers RESTful services which can also be used from other client applications. Users can directly submit and retrieve models from BioModels Database and the CellML Model Repository, two open repositories of systems biology models.

Taken together, our tools support researchers in creating and exploring COMBINE archives.



OMEX Meta Data

The COMBINE archive specification is a highly extensible container format and uses the RDF/XML standard to annotate content with different types of meta data. One of these meta types is OMEX. OMEX provides basic information about a model's provenance, by holding data about the author(s), time of creation and time of modifications. To keep things simple and lightweight, the OMEX meta data does not supply any mechanism for version control, although history tracking can be easily archived by using a version control system [7].

References

- [1] <http://co.mbine.org/home>
- [2] Le Novère et al.: Meeting report from the first meetings of the Computational Modeling in Biology Network (COMBINE). *Standards in genomic sciences*, 2011.
- [3] Bergmann et al.: COMBINE archive: One File To Share Them All. *arXiv*, 2014.
- [4] <https://sems.uni-rostock.de/cat>
- [5] <https://sems.uni-rostock.de/trac/combinearchive>
- [6] <http://webcat.sems.uni-rostock.de>
- [7] Waltemath et al.: Improving the reuse of computational models through version control. *Bioinformatics*, 2013.



Martin Scharm, Florian Wendland,
Martin Peters, Markus Wolfien,
Tom Theille, Dagmar Waltmath
<http://www.sbi.uni-rostock.de>



Identifying, Interpreting, and Communicating Changes in XML-encoded Models of Biological Systems

Universität
Rostock



Traditio et Innovatio



SYSTEMS BIOLOGY
BIOINFORMATICS
ROSTOCK

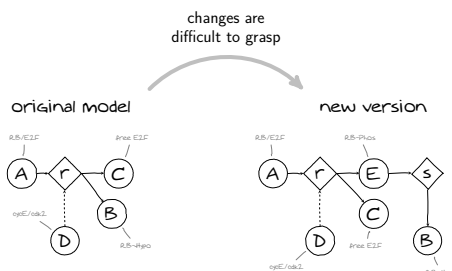
SEMS
simulation experiment management system

Summary

Even though models of biological systems are developed, revised, extended, and reproduced all the time, we lack methods to track the evolution of these models.

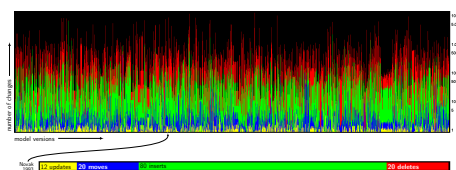
We here present tools for version control for computational models, which help to:

- Gain insights in the modelling process, thereby increasing confidence in models [3].
- Improve reproducibility of model-based scientific results in life science.
- Extend, improve, and correct existing models.
- Find related models.



Motivation

The data from the CellML Model Repository [9] and BioModels Database [10] prove that models change a lot over time. The image below shows the differences between the versions of all models available from the CellML Model Repository. Entities are mostly inserted or deleted, but often the document structure is modified by moving subtrees around the document.

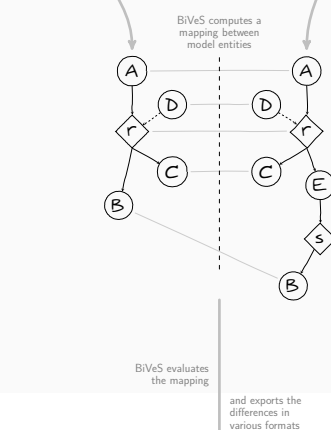


BiVeS

BiModel Version Control System [1]:

- Java library to map hierarchically structured content.
- Compares models encoded in standardised formats (currently: **CellML** [4] and **SBML** [5]).
- Matches unchanged or moved entities in model documents.
- Identifies "inserts" and "deletes".
- Constructs a diff (in XML format).
- Interprets identified changes and filters for biologically and mathematically relevant differences.

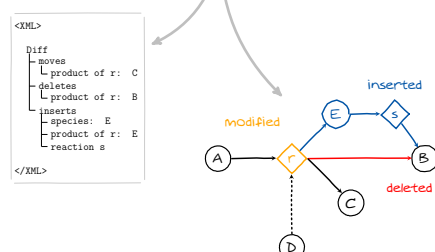
The diff, produced by BiVeS, can be used to grasp the changes which occurred between two versions of a model.



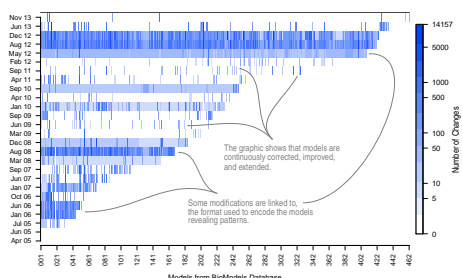
BudHat

The BudHat [2] web interface demonstrates the capabilities of BiVeS, by displaying the detected differences in various formats:

- The XML-encoded delta.
- A reaction network highlighting the changes using CytoscapeWeb [6], CytoscapeJS [7], or Graphene [8].
- A human readable report listing all modifications relevant to the model.



Results



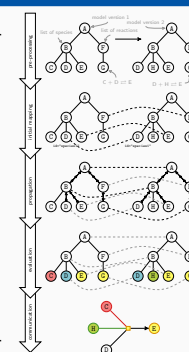
We developed an algorithm that identifies the differences between versions of a computational model and implemented it in our Java library BiVeS. Using BiVeS we analysed the versions of publicly available models. Models are subject to continual modifications, as seen in the figure on the left, and we recognised a pattern in the updates. We distinguish between model related modifications (corrections, improvements, extensions) and format specific updates. Taken together, our solution provides novel insights into the evolution of computational models. For the first time we are able to automatically compare computational models and export the differences in human readable formats.

BiVeS's Algorithm

After the document trees have been pre-processed, the algorithm proceeds in three major steps:

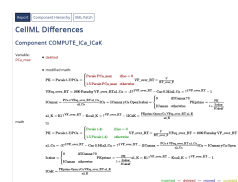
- Identical entities are mapped onto each other.
- This initial mapping is propagated into the tree.
- The resulting mapping is evaluated and modifications are classified.

In addition, BiVeS is able to understand the impact of detected modifications and exports the differences in various formats, such as the highlighted chemical reaction network or a text based summary. These outputs can easily be integrated in existing tools to increase the benefit for the users.



Integration of BiVeS

There are three different ways of integrating BiVeS:



The Functional Curation Project [11] uses BiVeS's Web Service to track the modifications of models on their website.



The SEEK platform [12] uses BiVeS's command line interface to compare the models in the database.

BiVeS provides a sophisticated API to compare models and then visualise the results. This API is, for example, used by BudHat.

Discover the benefits yourself at budhat.sems.uni-rostock.de/!

References

- [1] <http://sems.uni-rostock.de/bivess/>
- [2] <http://sems.uni-rostock.de/budhat/>
- [3] Wallerath et al.: Improving the reuse of computational models through version control Bioinformatics, 2013
- [4] Hucka et al.: The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models Bioinformatics, 2003
- [5] Lippert et al.: CellML: Its future, present and past Progress in Biophysics and Molecular Biology, 2004
- [6] Lippert et al.: Cytoscape Web: an interactive web-based network browser Bioinformatics, 2010
- [7] <http://cytoscape.github.io/cytoscape.js/>
- [8] <http://stanleygo.com/graphene-sems/>
- [9] Lippert et al.: The CellML Model Repository Bioinformatics, 2008
- [10] Li et al.: BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models BMC Systems Biology, 2010
- [11] Cooper et al.: High-throughput functional curation of cellular electrophysiology models Progress in Biophysics and Molecular Biology, 2011
- [12] Wöhrmann et al.: The SEEK platform for sharing data and models in systems biology Methods, 2011



Martin Scharm
University of Rostock
sems.uni-rostock.de/scharm
martin.scharm@uni-rostock.de



Dagmar Waltemath
University of Rostock
sems.uni-rostock.de/waltemath
dagmar.waltemath@uni-rostock.de



Olaf Wolkenhauer
University of Rostock
sems.uni-rostock.de/wolkenhauer
olaf.wolkenhauer@uni-rostock.de



THE
ROYAL
SOCIETY

EPSRC

Engineering and Physical Sciences
Research Council