Exploiting Explicit Context Information for the Automatic Generation of Simulation Experiments

Dissertation

zur

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

vorgelegt von

Pia Wilsdorf,
geb. am 19.08.1993
in Dresden

Rostock, 31. Juli 2024

https://doi.org/10.18453/rosdok_id00004800

Gutachter:

- Prof. Dr. Adelinde M. Uhrmacher (Universität Rostock)
- Prof. Dr. Jane Hillston (University of Edinburgh)
- Dr. Nikolas Popper (Technische Universität Wien)

Datum der Einreichung: 31. Juli 2024
Datum der Verteidigung: 14. November 2024

Abstract

Modeling and simulation have been established as indispensable tool in science and engineering, offering means to explore and analyze complex systems, thereby enhancing the prediction and understanding of real-world phenomena. Developing valid models that accurately reproduce these phenomena is an intricate process, involving steps of model building intertwined with various simulation experiments for model calibration, validation, and analysis.

Especially conducting simulation experiments is challenging as their specification and execution vary depending, for example, on the diverse experiment types, methods, and tools required. This dissertation addresses the central question of how to support modelers in this intricate process, specifically, by automatically generating and executing simulation experiments. A pivotal strategy identified for achieving this goal is the generation of simulation experiments through reuse and adaption. The strategy can be subdivided into three parts, and the dissertation is organized accordingly.

The first part contributes to the explicit and automated specification of simulation experiments, employing model-driven engineering. In this approach, metamodels are crucial for formalizing the central ingredients of a simulation experiment and for defining a domain-specific language that allows representing simulation experiments in a tool-agnostic manner. This enables the automatic generation, transformation, and adaption of executable experiment specifications via a model-driven engineering pipeline.

The second part underlines the importance of context for understanding and conducting simulation studies, and particularly simulation experiments. A comprehensive and formalized definition of the conceptual model is introduced, which subsumes typical early-stage artifacts of a simulation study, from research objectives and assumptions to information sources, enriched with metadata. The conceptual model can be complemented by provenance, which is concerned with the process of generating and using the various artifacts. Formal approaches for representing provenance information are discussed that can illuminate the entire story of a simulation study, including the interrelations between artifacts, activities of model building and simulation experiments, and related studies. Open model databases are investigated as a key means for distributing, editing, and interconnecting provenance graphs and the conceptual model, ensuring their computational accessibility.

The third part focuses on utilizing the formal, machine-accessible representations to facilitate the automatic interpretation and exploitation of context information. This context information is integrated with the model-driven approach for experiment specification and execution to form the key components of the Reuse and Adapt framework for Simulation Experiments (RASE). In this framework, provenance patterns and inference rules enable the automatic initiation of an experiment generator depending on the last activities of a modeler, automatic identification of earlier simulation experiments to reuse, and their adaption to the context of a new simulation model. This supports the iterative nature of the modeling and simulation life cycle, enabling scenarios such as automatic regression testing and cross-validation of related models.

The results showcase how modelers can benefit from the developed approaches, emphasizing systematic, effective, and methodologically rigorous simulation studies while preserving user flexibility and control. Demonstrations in open-source software prototypes and diverse case studies, spanning stochastic discrete-event simulations, virtual prototyping, finite element analysis, and agent-based simulations, highlight the versatility and applicability of the proposed framework.

Finally, the dissertation identifies new research avenues towards fully automating simulation experiments as well as the entire modeling and simulation life cycle.

Zusammenfassung

Modellierung und Simulation haben sich als unverzichtbare Werkzeuge in Wissenschaft und Technik etabliert. Sie bieten Mittel zur Erforschung und Analyse komplexer Systeme und verbessern so die Vorhersage und das Verständnis von Phänomenen der realen Welt. Die Entwicklung valider Modelle, die diese Phänomene genau reproduzieren können, ist ein komplexer Prozess, der Schritte der Modellbildung umfasst, die mit verschiedenen Simulationsexperimenten zur Modellkalibrierung, -validierung und -analyse verflochten sind.

Insbesondere die Durchführung von Simulationsexperimenten stellt eine Herausforderung dar, da deren Spezifikation und Ausführung variiert, z. B. in Abhängigkeit von den verschiedenen Experimenttypen, Methoden und Werkzeugen, die benötigt werden. Diese Dissertation beschäftigt sich mit der zentralen Frage, wie Modelliererinnen und Modellierer in diesem komplexen Prozess unterstützt werden können, insbesondere durch die automatische Generierung und Ausführung von Simulationsexperimenten. Eine zentrale Strategie zur Erreichung dieses Ziels ist die Generierung von Simulationsexperimenten durch Wiederverwendung und Adaption. Die Strategie kann in drei Ansätze unterteilt werden, und die Dissertation ist entsprechend gegliedert.

Der erste Ansatz trägt zur expliziten und automatisierten Spezifikation von Simulationsexperimenten bei, indem er Methoden der modellgetriebenen Entwicklung einsetzt. In diesem Ansatz sind Metamodelle entscheidend für die Formalisierung der zentralen Bestandteile eines Simulationsexperiments und für die Definition einer domänenspezifischen Sprache, die es erlaubt, Simulationsexperimente in einer werkzeugunabhängigen Weise darzustellen. Dies ermöglicht die automatische Generierung, Transformation und Anpassung von ausführbaren Experimentspezifikationen über eine modellgetriebene Entwicklungspipeline.

Der zweite Ansatz unterstreicht die Bedeutung von Kontext für das Verständnis und die Durchführung von Simulationsstudien, insbesondere von Simulationsexperimenten. Es wird eine umfassende und formalisierte Definition des konzeptionellen Modells vorgestellt, das typische Artefakte der frühen Phase einer Simulationsstudie umfasst, von Forschungszielen und Annahmen bis hin zu Informationsquellen, angereichert mit Metadaten. Das konzeptionelle Modell kann durch Provenienz ergänzt werden, welche sich mit dem Prozess der Erzeugung und Verwendung der verschiedenen Artefakte befasst. Es werden formale Ansätze zur Darstellung von Provenienz-Informationen diskutiert, die die gesamte Geschichte einer Simulationsstudie repräsentieren können, einschließlich der Beziehungen zwischen Artefakten, Aktivitäten der Modellbildung und Simulationsexperimenten sowie verwandten Studien. Offene Modelldatenbanken werden als zentrales Mittel zur Verteilung, Bearbeitung und Vernetzung von Provenienz-Graphen und dem konzeptionellen Modell untersucht, um deren maschinelle Zugänglichkeit zu gewährleisten.

Der dritte Ansatz konzentriert sich auf die Nutzung der formalen, maschinell zugänglichen Darstellungen, um die automatische Interpretation und Verwertung von Kontextinformationen zu erleichtern. Diese Kontextinformationen werden mit dem modellgetriebenen Ansatz für die Spezifikation und Ausführung von Experimenten integriert und bilden die Schlüsselkomponenten des Reuse and Adapt Framework for Simulation Experiments (RASE). In diesem Rahmen ermöglichen Provenienz-Patterns und Inferenzregeln die automatische Initiierung eines Experimentgenerators in Abhängigkeit von den letzten Aktivitäten einer Modelliererin oder eines Modellierers, die automatische Identifizierung früherer Simulationsexperimente zur Wiederverwendung und deren Anpassung an den Kontext eines neuen Simulationsmodells. Dies unterstützt die iterative Vorgehensweise im Modellierungs- und Simulationslebenszyklus und ermöglicht Szenarien wie automatische Regressionstests und Kreuzvalidierung verwandter Modelle.

Die Ergebnisse zeigen, wie Modelliererinnen und Modellierer von den entwickelten Ansätzen

profitieren können, wobei der Schwerpunkt auf systematischen, effektiven und methodisch präzisen Simulationsstudien liegt und gleichzeitig die Flexibilität und Kontrolle der Benutzerinnen und Benutzer erhalten bleibt. Demonstrationen in Open-Source-Software-Prototypen und verschiedenen Fallstudien, welche stochastische diskret-ereignisorientierte Simulationen, virtuelles Prototyping, Finite-Elemente-Analyse und agentenbasierte Simulationen umfassen, unterstreichen die Vielseitigkeit und Anwendbarkeit des vorgestellten Frameworks.

Abschließend identifiziert die Dissertation neue Forschungsrichtungen zur vollständigen Automatisierung von Simulationsexperimenten sowie des gesamten Modellierungs- und Simulationslebenszyklus.

Acknowledgements

First and foremost, I want to thank my supervisor, Adelinde Uhrmacher, for her unwavering support, guidance, and belief in the significance of this work. Her mentorship and the truly productive environment she created were pivotal to my academic development and the successful completion of this research.

I would like to thank my colleagues of the modeling and simulation group, including Andreas Ruscheinski, Tom Warnke, Kai Budde, Fiete Haack, Oliver Reinhardt, Philipp Henning, Maria Pierce, and Till Köster for their constructive feedback, cooperation, and numerous discussions. I am grateful to my collaborators Julius Zimmermann, Jason Hilton, and Jakob Heller for providing the case studies and for their contributions to the corresponding research papers.

I would also like to thank all the students that assisted me in the various projects: Marcus Dombrowsky, Anja Wolpers, Nadine Fischer, Marian Zuska, Glenn Skrzypczak, and Anton Willy Kirchhübel.

I want to thank my parents and grandmother for their support during the past six years. Most importantly, I am grateful to my partner, Michael Rethfeldt, for his patience and for always believing in me.

Contents

1 Introduction **1**
- 1.1 Contributions . . . 3
- 1.2 Case Studies . . . 4
  - 1.2.1 Stochastic Model of the Wnt Signaling Pathway . . . 4
  - 1.2.2 Virtual Prototyping of a Neurostimulator . . . 6
  - 1.2.3 Finite Element Analysis of Electric Fields . . . 7
  - 1.2.4 Agent-based Model of Human Migration . . . 8
- 1.3 Outline . . . 9
- 1.4 Bibliographical Note . . . 9

2 Background and Objective **13**
- 2.1 The Modeling and Simulation Life Cycle . . . 13
  - 2.1.1 Research Questions of a Simulation Study . . . 13
  - 2.1.2 Steps of the Life Cycle . . . 14
  - 2.1.3 Specifics of Modeling and Simulation Approaches . . . 17
- 2.2 Simulation Experiments in the Modeling and Simulation Life Cycle . . . 18
  - 2.2.1 Roles of Simulation Experiments . . . 19
  - 2.2.2 Structure of a Simulation Experiment . . . 19
  - 2.2.3 Types of Simulation Experiments . . . 21
- 2.3 Tools for Simulation Experiments . . . 34
- 2.4 Computerized Support for Simulation Experiments . . . 37
- 2.5 Objective and Outline . . . 38
- 2.6 Summary . . . 39

3 Making Simulation Experiments Explicit **41**
- 3.1 Motivation . . . 41
- 3.2 State of the Art . . . 43
  - 3.2.1 Explicit Experiment Specification . . . 43
  - 3.2.2 Model-driven Engineering and Generation of Simulation Experiments . . . 45
- 3.3 Model-driven Approach for Conducting Simulation Experiments . . . 46
  - 3.3.1 Framework Overview . . . 46
  - 3.3.2 Metamodeling Language . . . 46
  - 3.3.3 Specification and Composition . . . 48
  - 3.3.4 Metamodel Repository . . . 50
  - 3.3.5 Interfaces . . . 51
  - 3.3.6 Experiment Validation . . . 52
  - 3.3.7 Bindings . . . 52
- 3.4 Implementation . . . 54
- 3.5 Evaluation . . . 55
  - 3.5.1 Improved Knowledge Sharing across Domains . . . 56
  - 3.5.2 Increasing Productivity and Quality for Complex Experiments . . . 61
  - 3.5.3 Reusability . . . 65
  - 3.5.4 Automation . . . 67
- 3.6 Summary . . . 69

4 Making the Context of a Simulation Study Explicit 71
4.1 Motivation . . . 71
4.2 Making Early-Stage Products Explicit . . . 72
4.2.1 State of the Art . . . 73
4.2.2 Case Study . . . 80
4.2.3 An Integrated Conceptual Model Definition . . . 80
4.2.4 Implementation . . . 90
4.2.5 Discussion . . . 91
4.3 Making Relations between Products, Activities and Studies Explicit . . . 92
4.3.1 State of the Art . . . 94
4.3.2 Case Study . . . 103
4.3.3 Representing Model Databases as Provenance Graphs . . . 104
4.3.4 Implementation . . . 110
4.3.5 Exploring and Exploiting the Generated Provenance Graph . . . 111
4.3.6 Discussion . . . 115
4.4 Summary . . . 117

5 Generating Simulation Experiments by Reuse and Adaption 119
5.1 Motivation . . . 119
5.2 State of the Art . . . 121
5.3 Typical Reuse Scenarios in Simulation Studies . . . 124
5.3.1 Repeated Model Validation after Model Extension and Composition . . . 125
5.3.2 Repeated Model Calibration . . . 125
5.3.3 Repeated Model Analysis . . . 125
5.3.4 Cross-Validation with Related Simulation Studies . . . 126
5.3.5 Comparison of Alternative Implementations . . . 126
5.3.6 Synchronization of Concurrently Developed Models . . . 126
5.4 Reuse and Adapt Framework for Simulation Experiments . . . 126
5.4.1 Architecture . . . 127
5.4.2 Experiment Reuse by Provenance Graph Transformation Rules . . . 128
5.4.3 Rule Matching . . . 132
5.4.4 Adaption of the Experiment Specifications . . . 135
5.5 Implementation . . . 138
5.6 Case Studies . . . 140
5.6.1 Configuration of the Rule Set . . . 140
5.6.2 Repeated Analysis of a Migration Model . . . 141
5.6.3 Cross-Validation of two Models of the Wnt/β-Catenin Signaling Pathway . . . 143
5.6.4 Results . . . 145
5.7 Discussion . . . 146
5.8 Generating Simulation Experiments from Scratch . . . 148
5.8.1 Central Challenges . . . 149
5.8.2 Scenarios for Generating “from Scratch” . . . 152
5.9 Summary . . . 154

6 Conclusions 155
6.1 Summary . . . 155
6.2 Outlook . . . 156

A Metamodels of Simulation Experiments 159

B Excursus: Ontology of Sensitivity Analysis Methods 169

C Statistical Model Checking Experiment in SESSL **173**

Bibliography **175**

List of Figures

1.1 The two strategies for experiment generation: from scratch and reuse & adapt. . 3
1.2 Schematic of the Wnt/β-catenin model. 5
1.3 SystemC-AMS model of the neurostimulator STELLA 2.0. 6
1.4 CAD model and mesh of the electrical stimulation chamber. 7
1.5 Schematic of the migration model. 8

2.1 The modeling and simulation life cycle. 16
2.2 Structure of a simulation experiment that varies the model parameters. 20
2.3 Transient and steady state behavior of a simulation. 22
2.4 2^k full factorial design and 2^{k-1} fractional factorial design. 23
2.5 Overview of a sensitivity analysis. 25
2.6 Two procedures for parameter estimation. 27
2.7 Simulation-based optimization loop. 29
2.8 Workflow of a statistical model checking experiment. 30
2.9 Process of planning, running, and evaluating what-if simulations. 32
2.10 Error and convergence control feedback loop. 33
2.11 Modeling and simulation life cycle with automatic experiment generation. 39

3.1 Four-layered MDE architecture for simulation experiments 47
3.2 The base metamodel of Listing 3.1 defined in UML. 49
3.3 Dependency between two experiment metamodels defined by a weaving model. . 51
3.4 Screenshot of the experiment generation GUI. 52
3.5 Bindings to modeling and simulation systems and analysis tools. 54
3.6 Schematic of the generated code for the three SA experiments. 62
3.7 First- and total-order Sobol indices calculated for the three models. 63
3.8 Results of the cross-validation of the Lee et al. model and the Haack et al. model. 67
3.9 The conceptual model artifact of the artifact-based workflow. 68
3.10 Convergence of the current with respect to the number of degrees of freedom. . . 69

4.1 The spectrum of conceptual model definitions. 75
4.2 Data sources used in the Wnt signaling study. 89
4.3 The Conceptual Modeling Wiki. 91
4.4 PROV-DM graphical notation. 98
4.5 Various views on an exemplary provenance graph. 100
4.6 The BioModels page of the simulation study by Padala el al. 104
4.7 Filling the definition of the qualitative model. 108
4.8 Provenance graph(s) created from the BioModels entry BIOMD0000000652. . . . 111
4.9 Query result showing all the simulation model versions. 112
4.10 Query result showing the curating simulation model activity. 113
4.11 Query result showing all simulation model extensions and compositions. 114
4.12 Query result showing all analyzing simulation model activities. 114
4.13 Provenance graph of the Wnt signaling simulation study by Haack et al. 115

5.1 Overview of the reuse and adapt framework for simulation experiments (RASE). 127
5.2 Provenance patterns used in the approach. 129
5.3 Rule matching at a given activity a. 133

5.4 Excerpt of an SBML file. 136
5.5 Adapting and generating simulation experiments based on metamodels. 137
5.6 API for capturing provenance. 139
5.7 A reuse rule for the repetition of sensitivity analysis. 140
5.8 A reuse rule for cross-validation. 141
5.9 Provenance graph showing the generation of a sensitivity analysis. 142
5.10 Sensitivity indices of the original and generated experiments. 143
5.11 Provenance graph showing the generation of a cross-validation experiment. 144
5.12 Results of the cross-validation experiment. 146
5.13 Provenance patterns for primary and secondary data collection. 148
5.14 Scenarios for generating simulation experiments from scratch. 152

B.1 Ontology of sensitivity analysis methods implemented with Protégé. 171

List of Tables

2.1 Modeling and simulation frameworks and experiment specification languages. . . 35
2.2 Selection of tools and libraries for conducting specific experiment types. 36

3.1 Base metamodel for conducting experiments using discrete-event simulation. . . . 57
3.2 Modified metamodel component for virtual prototyping of heterogeneous systems. 59
3.3 Metamodel for the experiment type “global sensitivity analysis”. 60

4.1 Structure and contents of TRACE documents (elements 1–4). 77
4.2 Structure and contents of TRACE documents (elements 5–8). 95
4.3 Provenance entities recognized for the different model versions. 107
4.4 Types of provenance activities 110

5.1 Comparison of related work on generating simulation experiments. 123
5.2 Table of variables enhanced with ontology tags. 135
5.3 Adaptions based on context information. 138
5.4 UniProt tags used to identify and map the species involved in the two models. . . 145

A.1 Base metamodel for finite element analysis. 160
A.2 Metamodel for the experiment type “steady state estimation”. 161
A.3 Metamodel for the experiment type “parameter scan”. 162
A.4 Metamodel for the experiment type “local sensitivity analysis”. 163
A.5 Metamodel for the experiment type “parameter estimation”. 164
A.6 Metamodel for the experiment type “optimization”. 165
A.7 Metamodel for the experiment type “statistical model checking”. 166
A.8 Metamodel for the experiment type “what-if analysis”. 167
A.9 Metamodel for the experiment type “convergence testing”. 168

List of Code Listings

3.1 Base metamodel of a fictive modeling domain defined in JSON Schema. 50
3.2 Simulation experiment filled according to the JSON metamodel. 53
3.3 Cross-validation experiment in the exchangeable JSON format. 66

B.1 Specification of the class “Morris” of the SA method ontology. 170
B.2 DL query for finding all suitable SA methods for the current simulation model. . 170

C.1 Statistical model checking experiment in SESSL for cross-validation. 173

1 Introduction

Modeling and simulation (M&S), alongside empirical evaluation and analytical reasoning, is regarded to be the third pillar of science [1]. It provides additional insights that are often not possible based on real-world experiments or theoretical analysis alone. This may be due to the fact that some real-world experiments would be too technically challenging, financially expensive, or morally unacceptable, or that closed-form solutions do not exist for some problems. Therefore, in many domains—be it in science, engineering, or industry—simulation studies have become indispensable for understanding, designing, predicting, and decision-making in solving complex questions.

M&S studies are intricate processes where cycles of model refinement and extension are closely intertwined with diverse simulation experiments for model calibration, validation, or analysis [2, 3]. Starting with a research question about the system of interest, simulation models are successively refined based on assumptions and simplifications. Along the way, requirements or hypotheses referring to outputs of the model are specified, and data sources are identified that may be used to initialize the model or as input for simulation experiments used for validation or calibration of the simulation model until a meaningful yet concise approximation of the system has been achieved [4, 5]. The complex process of conducting a simulation study, therefore, requires both science and art [6]. The programming, statistical analysis, and other methodological aspects represent the scientific part, whereas the art refers to coming up with the model components, interpreting the (intermediate) results, and making decisions, for example, about what alternative scenarios to evaluate.

The increasing popularity and importance of simulation [7, 8] is accompanied by an increasing awareness that many simulation results cannot be relied upon. For instance, literature reviews showed that a significant number of authors do not provide sufficient documentation of the simulation tools used and simulation results obtained [9]. Other studies showed that even if documented, large deviations between observed data of the modeled system and data generated by simulation can occur, as well as large deviations between the results of independent modelers, presumably due to the different choices of methods and equations [10]. This is known as the reproducibility and credibility crisis of simulation [11]. Simulation, as well as the sciences in general, consequently face the challenge of finding new remedies for handling the uncertainties and risks associated with scientific procedures and results [12]. This comprises the questions of how fast useful models can be developed and analyzed, how the achieved results can be interpreted and reused, and how results and crucial aspects of simulation studies can be communicated with domain experts and decision makers. Much of this is particularly difficult, due to simulation studies being knowledge-intensive processes, in which the knowledge and expertise are often implicit, held within the minds of experts, or take the form of informal best practices or guidelines [13]. In addition, simulation studies can be characterized as data-driven processes [14], implicating that the trust in the models is also directly related to the trust in the data used as input, the simulation experiments and analyses conducted in the simulation study as well as the outputs generated, and whether those adequately address the research question. Simulation experiments therefore massively contribute to the reproducibility and credibility of a simulation study, as they are the centerpiece between models, input data, and simulation results, and thereby expedite important activities, such as the calibration, validation, and analysis of models.

A simulation experiment is defined to be an "experiment with a model" [15, page 5], where "[a]n experiment is the process of extracting data from a system by exerting it through its inputs" [15, page 4]. Despite this simple definition, in practice, simulation experiments are more elaborate.

Conducting a simulation experiment involves several tasks including the configuration of the model parameters, execution of the model, retrieval of observations, analysis of the observed data, and evaluation and selection of further parameter configurations [16]. The complexity of conducting a simulation experiment is reinforced by the various types of simulation experiments, the plethora of methods available for sampling the parameter values and analysis of the model output (which requires sufficiently many replications in case of a stochastic model), the diverse M&S approaches and metaphors, and the range of tools that modelers are expected to master [17].

Due to the wide range of expertise needed [6] and since simulation experiments present an important yet complicated part of simulation studies [18], an increasing number of approaches aim at assisting modelers in conducting their simulation experiments [19]. Within the last years, one major concern of developments has been to effectively exploit distributed computing resources and thus to allow efficient execution of large-scale simulation experiments [20, 21, 22]. Another focus of developments has been on suitable means to unambiguously and flexibly specify simulation experiments [23, 24, 25, 26]. The research presented in this dissertation builds on the latter but goes one step further by asking how to generate such simulation experiment specifications and execute them automatically. Generating the simulation experiments automatically will enable modelers to work in a more effective and systematic manner, save them time and reduce errors in applying the different methods [19]. Automation may even broaden the scope of analyses done with simulation models, and facilitate model and experiment reuse within the same or across related simulation studies.

To achieve situation-specific support of simulation experiments, context about the simulation study needs to be accessible, unambiguously specified, and thus automatically interpretable. Various types of knowledge have to be integrated and put into relation, including: a) knowledge about the structure of different types of simulation experiments, b) knowledge about the methods to be applied and the tools available, c) knowledge about the life cycle of simulation studies, d) context about the simulation study at hand, such as all assumptions, requirements, and input data, e) context about the central model building activities, simulation experiments, and the simulation results produced, and f) information about the relation to other simulation studies. Items a)–d) put an emphasis on the conceptual modeling phase, which, in the M&S life cycle, takes place before the model building and refinements and is crucial for setting the context and determining the direction of the simulation study. In particular, the conceptual model describes the system of interest as such and all modeling related "objectives, inputs, outputs, content, assumptions, and simplifications" [27], and relates those to the analysis that shall be conducted based on the simulation model. A well-specified conceptual model integrates and provides a variety of requirements and information that is of assistance for conducting a simulation study [8]. For example, it determines which behavior the simulation model must show under which circumstances, which data it must reproduce, or which sensitivity shall be tolerated referring to parameter values. Next to the conceptual model, provenance will form another essential pillar of knowledge required for automatic experiment generation, see items e) and f). Provenance, as used in this dissertation, refers to an all-encompassing documentation that does not only focus on the simulation model as the final product but instead reflects the entire process of conducting a simulation study [28]. It, therefore, contains valuable information about how the various products of a simulation study, such as requirements or data, evolved and contributed to creating some model, experiment, or output data and the exact role they played.

Figure 1.1 illustrates the various tasks involved in automatically generating a simulation experiment and how the different types of knowledge are accessed and integrated during the process. Usually, one starts by identifying the user's goal at the current point in the simulation study. The goal may be either given by the user, for example, saying "I wish to validate the model" or "I wish to relate my new model to simulation study x ...", or derived from past user activities. From there one has to determine what kind of experiment to generate and execute next in order to satisfy the goal. Here, two strategies can be pursued. The first strategy (see

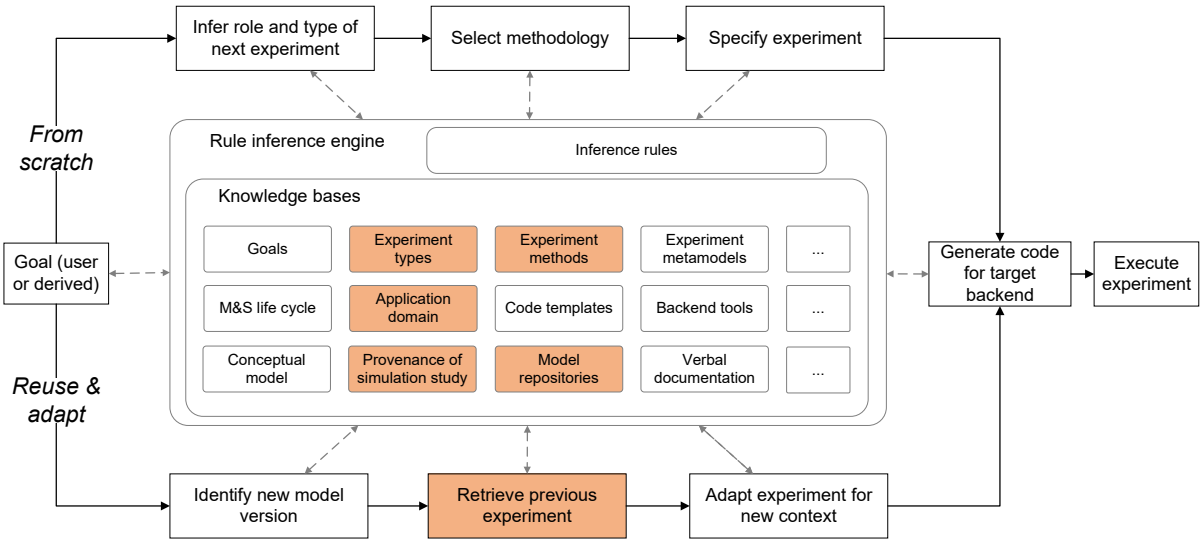


Figure 1.1: The two strategies for experiment generation: *from scratch* and *reuse & adapt*. Their various subtasks require different kinds of knowledge to be integrated via inference rules. The task of retrieving previous simulation experiments, for example, may consult provenance of the respective simulation study, related model repositories, and knowledge about the application domain as well as experiment types and methods (colored boxes).

Figure 1.1, upper branch) refers to building simulation experiments from scratch by inferring the next experiment type and the appropriate methods from the context, which may involve further knowledge, e.g., about the application domain. Once the experiment type, method, and simulation tool have been selected, corresponding metamodels can guide the collection of relevant information, templates can be filled from available context information, and finally, the experiment code can be generated and executed. In contrast, the second strategy (see Figure 1.1, lower branch) refers to reusing and adapting existing simulation experiments from the same or related simulation studies. This accounts for the fact that models and experiments are rarely built from scratch and leverages the iterative nature of simulation studies where the same goals (e.g., to test the model's sensitivity or to check some requirement) frequently reappear, and thus simulation experiments are re-applied in new contexts, for example, in light of a new model version after model extension or additional validation data.

This dissertation focuses on the second strategy and presents the process and ingredients required for automatically generating simulation experiments by reuse and adaption, in particular, by facilitating provenance with rich metadata and inference rules. This also lays the foundation for generating simulation experiments from scratch, as information like explicit experiment specifications and tool bindings are necessities for both strategies.

1.1 Contributions

This dissertation explores the automatic generation of simulation experiments by reuse and adaption based on provenance and other context information about a simulation study. The main contributions are the following:

- An overview of experiment types, their ingredients, methodologies and tooling, and classification regarding their roles in the simulation study. This forms the groundwork for defining metamodels of simulation experiments, ontologies of methods, and decision logic for generating the right kind of simulation experiment.

- A metamodeling approach for making the structure and ingredients of simulation experiments explicit, and a corresponding model-driven engineering pipeline that generates simulation experiments based on these metamodels. Actual metamodels for three simulation approaches and eight experiment types are provided.
- A discussion of what constitutes the context of a simulation study and the various ways in which context may be represented.
- A definition and data model for making the conceptual model explicit, including the research questions, assumptions, requirements, qualitative models, approaches, and data and information sources. The definition aims at a broad interpretation of the conceptual model, where the ingredients are structured in a knowledge graph and the information is semi-formal to allow machine processing while at the same time granting users as much flexibility as possible.
- Suggestions and means for integrating provenance graphs within model databases to allow making entire simulation studies, including their development and curation, explicit and queryable. This allows for improved findability, accessibility, interpretation, and comparison of models.
- A framework based on patterns defined in the provenance standard PROV-DM, which allow characterizing the central model building and model analysis activities. The provenance patterns serve as triggers for starting an automatic experiment generator and for identifying previous simulation experiments and other information to be reused and adapted in the current situation, e.g., for a new model iteration. The patterns are applied using transformation rules on the provenance graphs.
- A proposal for generating simulation experiments from scratch, i.e., without a previous simulation experiment as a blue-print and with only a loose documentation of the modeling process, and identification of the central challenges to be solved.
- Prototypical implementations of the presented approaches provided in open repositories.
- Conduction of four case studies, which demonstrate the versatility of the developed approaches regarding the used modeling metaphors, application domains, and tooling.

1.2 Case Studies

During this dissertation, case studies are used to illustrate the different concepts and show the developed tools at work. In particular, the simulation studies were selected to evaluate the usability of the developed approaches for a broad audience in M&S due to their versatility regarding the used tools, M&S approaches and application domains. In the following, the four case studies are briefly introduced in the order of their appearance.

1.2.1 Stochastic Model of the Wnt Signaling Pathway

The first case study refers to the canonical Wnt/β-catenin signaling pathway, which is a central pathway that regulates proliferation as well as differentiation of cells [29]. Deregulated forms of this pathway are involved in a number of human cancers and developmental disorders [30]. An in-depth understanding of the underlying cellular processes is crucial for developing targeted treatments.

Previous simulation studies have intensively studied the key intracellular mechanisms of the Wnt signaling pathway and the impact of external stimulation on the cell [32]. Those studies, however, largely neglected the importance of membrane-related processes. Therefore, a new Wnt

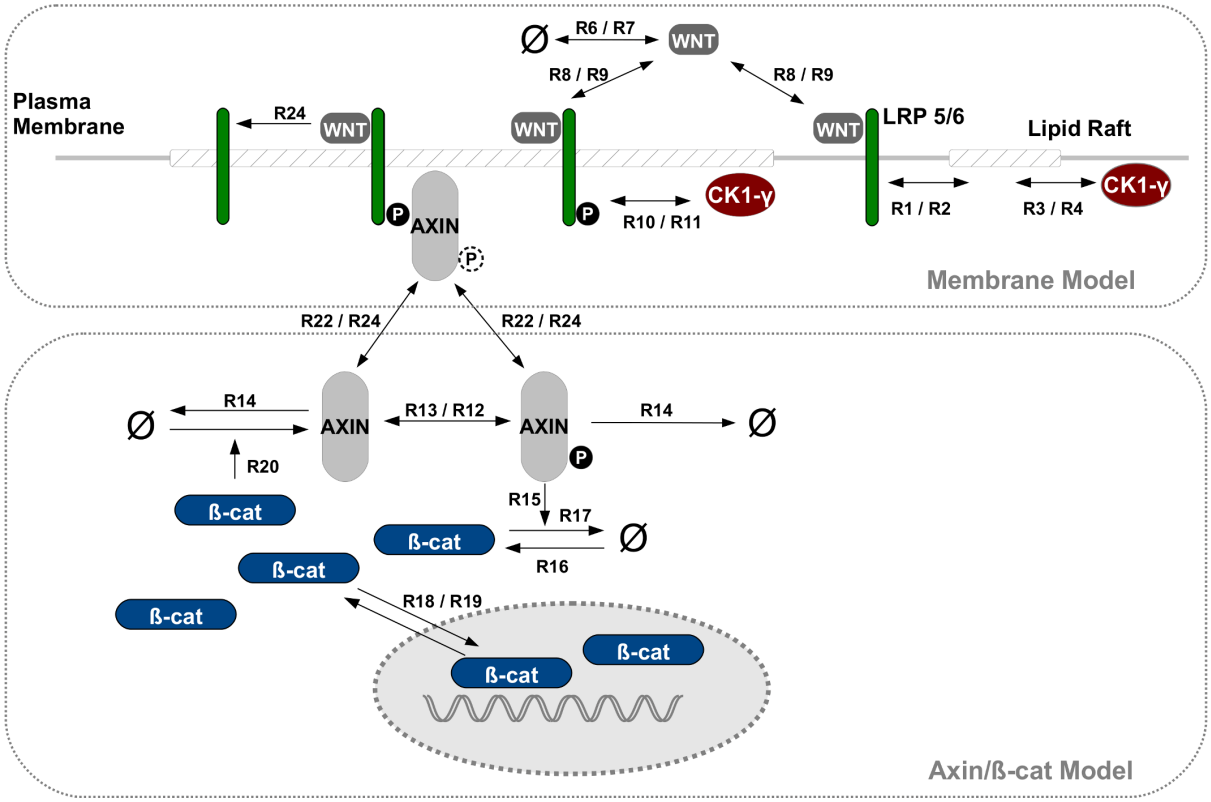


Figure 1.2: Schematic of the Wnt/β-catenin model combining membrane-related dynamics (top) with intracellular processes (bottom). Signal transduction is initiated by extracellular Wnt molecules via the receptors LRP5/6, followed by a cascade of reactions involving key proteins, such as Axin and CK1-γ, leading to an accumulation of β-catenin in the cytosol and finally the nucleus of the cell. Arrows indicate reactions between the model entities with designated reaction rate constants (R_i). Figure reused from [31].

simulation model is developed to combine intracellular and membrane-related dynamics [31]. Recent hypotheses and data put lipid rafts, specialized microdomains of the membrane, in the focus of research, as they have been found to sense the electric field and to direct cellular responses [33]. Thus, the Wnt model is extended accordingly to investigate the effect of both raft- and non-raft-associated endocytic processes of the Wnt/β-catenin receptor LRP6 in detail, including stochastic effects [34].

The new Wnt model is implemented in the rule-based multilevel modeling language ML-Rules [35]. This language uses the modeling metaphor of chemical reactions and, in addition, compartments to allow modeling and analyzing different organization levels (e.g., the membrane and the nucleus) of the cell. Figure 1.2 depicts the model with its two submodels, the intracellular axin/β-catenin model and the membrane model. Overall the Wnt model comprises 24 reactions, each with a reaction rate constant R_i. Simulation runs are carried out using the experimentation tool SESSL [36], which interfaces the stochastic simulation algorithm (SSA) [37] provided by ML-Rules. The Wnt model thus is a representative of the class of stochastic discrete-event simulation models [38, 39] with a continuous-time Markov chain semantics [40].

The canonical Wnt/β-catenin signaling pathway provides ample opportunity to demonstrate the specification and automatic generation of simulation experiments and the role of explicit context information. The pathway has been subject to numerous simulation studies [32, 41, 31, 34], many of which have been published in the BioModels database and therefore are accessible for further exploration and exploitation [42]. Detailed provenance has already been documented for some Wnt models (e.g., [34, 43]) to allow examining their interrelations, such as relation by extension,

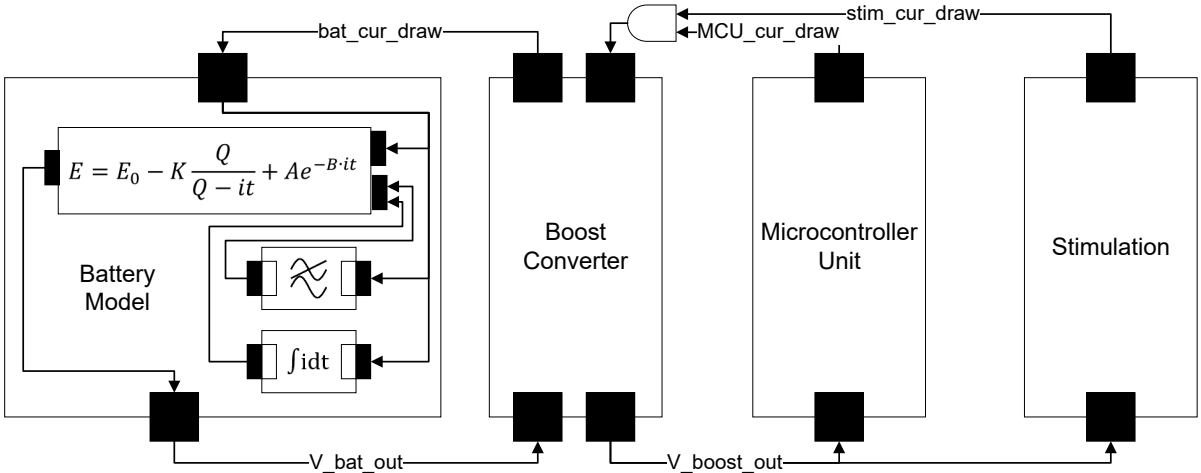


Figure 1.3: SystemC-AMS model of the neurostimulator STELLA 2.0, including the four sub-models for the boost converter, microcontroller unit (MCU), stimulation unit, and battery model (battery voltage depending on discharge). Note that the model uses the symbols suggested by the SystemC-AMS extensions User's Guide[1]. Redrawn with modifications from Figure 3 of [46].

revision, or cross-validation. Simulation models of the Wnt/β-catenin signaling pathway have also been the subject of review papers and analyses (e.g., [44, 45]), which highlights the interest in structured knowledge about that modeling domain as well as the need for methodological advancements.

1.2.2 Virtual Prototyping of a Neurostimulator

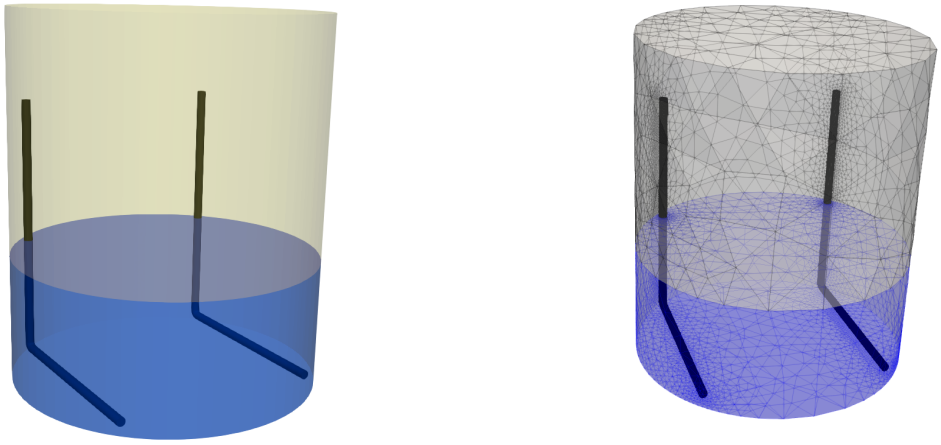
Deep brain stimulation is a therapy option for a multitude of neurological disorders. While it is widely implemented into the clinical routine, especially for Parkinson's disease and Dystonia, the underlying mechanisms are not fully understood [47]. Since most experiments cannot be performed directly on humans due to ethical reasons, small animal models, such as rodents, are necessary [48]. However, rodent-specific implants require a highly optimized level of miniaturization and low power consumption compared to human implants. With physical prototyping being infeasible, virtual prototyping is used to design optimized neurostimulators for rodents based on a computerized model and simulation experiments [46].

In this particular case study, a model is built to assist in the development of STELLA (SofTware dEfined impLantabLe modular plAtform), a device for preclinical deep brain stimulation [49]. The model considers the interplay between various heterogeneous system components, i.e., battery, boost converter, microcontroller, and stimulation unit (as illustrated in Figure 1.3). It thus combines digital as well as analog components. Whereas previous studies focused on the optimization of voltage levels inside the system, this study focuses on understanding the implications of different battery parameters on the overall runtime, as a multitude of batteries with different volume-runtime trade-offs are available for implants.

As the model includes components outside the digital domain (e.g., to model the voltage levels of a battery) it requires continuous-time representation in addition to discrete-time events. Thus, SystemC-AMS [50], an extension of the system description language SystemC, is used to allow for a hybrid M&S approach with both discrete and continuous signals. In addition, Python is used as a wrapper for starting simulation runs and processing the output data.

Due to its hybrid approach, the virtual prototyping study provides a case study for transferring

[1]SystemC AMS extensions user's guide, 2010. `https://www.accellera.org/images/downloads/standards/systemc/OSCI_SystemC_AMS_Users_Guide.pdf`, last accessed 19 July, 2024.



(a) CAD Model (b) Mesh

Figure 1.4: CAD model (a) and mesh (b) of the electrical stimulation chamber with the cell culture medium (blue), the two electrodes (black) and air (shaded). Figures reprinted from [52].

knowledge and concepts between the diverse application fields of M&S. Thus far, experiments in the context of virtual prototyping are typically specified and conducted in an ad-hoc manner. This is, e.g., manifested in the fact that in SystemC there is no clear separation between model, simulator, and experiment, contrary to what is recommended in the M&S theory [51]. Consequently, a more structured and systematic approach to virtual prototyping, and in particular to conducting analyses with the developed models, is welcome.

1.2.3 Finite Element Analysis of Electric Fields

Electrical stimulation is a promising tool for tissue engineering and regenerative medicine [53]. Prominent applications include deep brain stimulation [54] and bone regeneration [55]. However, an abundance of stimulation parameters (e.g., waveform, amplitude, and frequency of the stimulation signal as well as electrode material) needs to be considered, making it necessary to develop a profound understanding of the effect of electrical stimulation on cell tissue [56]. Numerical simulation studies can elucidate the electric field that cell experience and allow researchers to make informed design choices for novel electrical stimulation devices and the following in vitro and in vivo studies.

Finite element analysis (FEA) is a general numerical method that is capable of simulating complex geometries and accurately computing the properties and effects of electric fields. In this FEA study, a physical model given by partial differential equations describing electric fields in the frequency domain is solved. The 3D geometry of the system, i.e., the implant and its surrounding material, is modeled explicitly and corresponding boundary conditions and material properties are linked to the different domains of the geometric model.

The third case study applies FEA to compute the electric field distribution in a specific electrical stimulation chamber [57, 58]. Based on the computed field distributions, the biological response of the stimulated biological sample can be linked to certain specifications of the electrical stimulation setup, such as material properties or frequency of the stimulation signal [56]. The simulation study uses SALOME[2] to build the geometric model (CAD model) and to discretize it into a mesh with a finite number of elements. Figure 1.4 depicts the geometry consisting of a cell culture medium, air, and two electrodes, and the generated mesh. The physical model and the

[2] `www.salome-platform.org/`, last accessed 19 July, 2024.

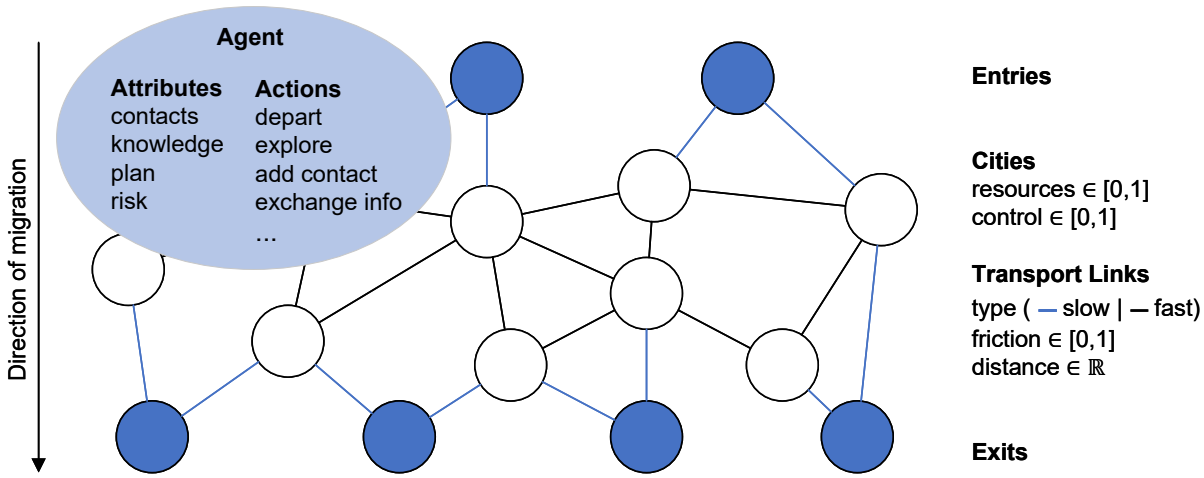


Figure 1.5: Schematic of the migration model. Human agents update their knowledge about the environment, their contacts and plan as they move through the route network from a set of designated entries to the exit nodes. Nodes represent cities, which are linked via slow and fast edges. Adapted from [63].

simulation experiments are specified and run using the Python package EMStimTools [59], which uses the FEniCS library [60] to solve the underlying equations.

Compared to discrete-event simulation, finite element analysis presents an entirely different kind of simulation. With the separation into a geometric and a physical model, new requirements and inputs for the support of these studies and their simulation experiments arise. The need for specialized support of FEA studies is revealed, e.g., by the development of targeted FEA workflows [52]. But also specialized simulation experiments are necessary, e.g., to validate the mesh of the geometric model [61]. Furthermore, reporting guidelines for FEA request the explicit and thorough documentation of these studies [62].

1.2.4 Agent-based Model of Human Migration

International migration affects not only the countries at both origin and destination but also the countries that are situated on common migration routes [64]. Thus, understanding migration flows and their consequences on the economy, culture, and politics is crucial for policymakers. Nevertheless, migration is still among the least well-understood demographic processes [65]. Especially the migration journey itself, including the individual's decisions and movements, has not received enough attention in existing studies [66].

Agent-based M&S has become an established tool for understanding and analyzing complex phenomena in demography and migration studies as it allows linking macro- and micro-level perspectives of complex processes [67]. This case study refers to an agent-based model of asylum migration to Europe [68, 69]. The model focuses on the formation of migration routes in response to the exchange of information (e.g., about nearby cities and their attractiveness). The agents in this model are characterized by their number of contacts, incomplete knowledge about the world, their current route plan that gets constantly updated when communicating with other agents, and their risk behavior. Another central component of this model is the migration route network, depicted in Figure 1.5. Agents attempt to move through the network to one of their target destinations while collecting information based on which they make their intelligent, cognitive decisions.

Two different approaches are taken to realizing this model. First, a model with discrete step-wise simulation semantics is implemented in Julia [70]. Second, to allow for more realistic behavior of the agents, a discrete-event model with continuous-time Markov chain semantics is

implemented, both in Julia and the domain-specific modeling language ML3, which was designed for modeling linked lives in demography [71]. The simulation experiments and analyses are carried out using the statistics tool R [72].

The model in this agent-based simulation study is successively refined and analyzed in response to developing new behavioral theories and the collection of additional data sources. These model iterations have explicitly been documented in [73], which provides a solid starting point for further investigation and characterization of the modeling and analysis process. Furthermore, this study offers ample opportunity to explore the impact of research questions, hypotheses, theories, and data on the next model iterations to be built and the simulation experiments to be conducted.

1.3 Outline

This dissertation is divided into six chapters. After the presentation of the problem, research objectives, and case studies in Chapter 1, Chapter 2 introduces the M&S life cycle, gives an overview of the types of simulation experiments, and surveys existing attempts at facilitating or automating those. Chapter 3 describes the design of a model-driven approach for conducting simulation experiments. Chapter 4 presents and discusses a variety of approaches for making the context of a simulation experiment, and the story of developing a simulation model explicit. Chapter 5 presents the RASE framework, which exploits the explicitly specified simulation experiments and the documented context information for reusing and adapting simulation experiments automatically. Finally, Chapter 6 summarizes the results and key insights of the dissertation and gives an outlook on future research topics.

1.4 Bibliographical Note

The concepts presented in this dissertation were developed as part of the DFG-funded research project "towards GeneRating and Executing Automatically Simulation Experiments" (GrEASE)[3], and parts of the dissertation have been published in journal articles and conference papers.

Chapter 3 is based on the content of the following publication about model-driven development of simulation experiments:

- **P. Wilsdorf**, J. Heller, K. Budde, J. Zimmermann, T. Warnke, C. Haubelt, D. Timmermann, U. van Rienen, and A. M. Uhrmacher. A model-driven approach for conducting simulation experiments. *Applied Sciences*, 12(16), 2022. doi:10.3390/app12167977.

The ideas from the earlier publication on experiment schemas have been incorporated in the concept of Chapter 3:

- **P. Wilsdorf**, M. Dombrowsky, A. M. Uhrmacher, J. Zimmermann, and U. v. Rienen. Simulation experiment schemas – beyond tools and simulation approaches. In *2019 Winter Simulation Conference (WSC)*, pages 2783–2794. IEEE, 2019. doi:10.1109/WSC40007.2019.9004710.

Chapter 3 is also inspired by work on using model documentation to fill templates. In the coauthorship of this paper, methodology, discussion of model documentation, and implementation of the case study were contributed:

- A. Ruscheinski, K. Budde, T. Warnke, **P. Wilsdorf**, B. C. Hiller, M. Dombrowsky, and A. M. Uhrmacher. Generating simulation experiments based on model documentations and templates. In *2018 Winter Simulation Conference (WSC)*, pages 715–726. IEEE, 2018. doi:10.1109/WSC.2018.8632515.

[3] `https://gepris.dfg.de/gepris/projekt/320435134?language=en`, last accessed 19 July, 2024.

A series of publications describes approaches for making information about a simulation study explicit, as well as approaches for exploiting that information. Parts of these publications were revised and reorganized for Chapter 4:

- **P. Wilsdorf**, F. Haack, and A. M. Uhrmacher. Conceptual models in simulation studies: Making it explicit. In *2020 Winter Simulation Conference (WSC)*, pages 2353–2364. IEEE, 2020. doi:10.1109/WSC48552.2020.9383984.

- **P. Wilsdorf**, N. Fischer, F. Haack, and A. M. Uhrmacher. Exploiting provenance and ontologies in supporting best practices for simulation experiments: A case study on sensitivity analysis. In *2021 Winter Simulation Conference (WSC)*, pages 1–12. IEEE, 2021. doi:10.1109/WSC52266.2021.9715362.

- **P. Wilsdorf** and A. M. Uhrmacher. Creating PROV-DM graphs from model databases. In *2022 Winter Simulation Conference (WSC)*, pages 2118–2129. IEEE, 2022. doi:10.1109/WSC57314.2022.10015331.

- **P. Wilsdorf**, A. W. Kirchhübel, and A. M. Uhrmacher. nbSimGen: Jupyter notebook extension for generating simulation experiments. In *2023 Winter Simulation Conference (WSC)*, pages 2884–2895. IEEE, 2023. doi:10.1109/WSC60868.2023.10408537.

A particularly noteworthy paper discusses the state of the art and open challenges in modeling and simulation. The author of this dissertation contributed literature reviews of formal approaches to modeling and approaches for automation, as well as discussions of future research directions. These contributions have been instrumental to writing parts of Chapters 4 and 6.

- A. M. Uhrmacher, P. Frazier, R. Hähnle, F. Klügl, F. Lorig, B. Ludäscher, L. Nenzi, C. Ruiz-Martin, B. Rumpe, C. Szabo, G. A. Wainer, and **P. Wilsdorf**. Context, composition, automation, and communication - the C2AC roadmap for modeling and simulation. *ACM Transactions on Modeling and Computer Simulation*, 34(4), 2024. doi:10.1145/3673226.

Chapter 5 is based on the following publication, which introduced the framework for reusing and adapting simulation experiments based on provenance patterns:

- **P. Wilsdorf**, A. Wolpers, J. Hilton, F. Haack, and A. Uhrmacher. Automatic reuse, adaption, and execution of simulation experiments via provenance patterns. *ACM Transactions on Modeling and Computer Simulation*, 33(1–2), 2023. doi:10.1145/3564928.

Section 5.7 reused parts of the following publication to illustrate further applications of the provenance patters:

- **P. Wilsdorf**, O. Reinhardt, T. Prike, M. Hinsch, J. Bijak, and A. M. Uhrmacher. Simulation studies of social systems – telling the story based on provenance patterns. *Royal Society Open Science*, 11(8):240258, 2024. doi:10.1098/rsos.240258

Chapter 5 closes with a discussion of ideas for generating simulation experiments from scratch, which is based on the ideas from the following publication:

- **P. Wilsdorf**, F. Haack, K. Budde, A. Ruscheinski, and A. M. Uhrmacher. Conducting systematic, partly automated simulation studies – unde venis et quo vadis. In *17th International Conference of Numerical Analysis and Applied Mathematics*, 2020. doi:10.1063/5.0026939.

Other published contributions were incorporated into the introduction, motivation, background and state-of-the-art sections of this dissertation. For the publication written as first author this includes languages for specifying requirements:

- **P. Wilsdorf**, M. Zuska, P. Andelfinger, A. M. Uhrmacher, and F. Peters. Validation without data - formalizing stylized facts of time series. In *2023 Winter Simulation Conference (WSC)*, pages 2674–2685. IEEE, 2023. doi:10.1109/WSC60868.2023.10408388.

For the publications written in co-authorship this includes the work on provenance ontologies for simulation studies, provenance capturing, and further applications of the conceptual model, provenance, and experiment generation. All of them are referenced in the body of this dissertation:

- A. Ruscheinski, **P. Wilsdorf**, M. Dombrowsky, and A. M. Uhrmacher. Capturing and reporting provenance information of simulation studies based on an artifact-based workflow approach. In *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, SIGSIM-PADS '19*, pages 185–196. ACM, 2019. doi:10.1145/3316480.3325514.

- K. Budde, J. Smith, **P. Wilsdorf**, F. Haack, and A. M. Uhrmacher. Relating simulation studies by provenance—developing a family of Wnt signaling models. *PLOS Computational Biology*, 17(8):1–27, 2021. doi:10.1371/journal.pcbi.1009227.

- T. Meyer, A. Ruscheinski, **P. Wilsdorf**, and A. M. Uhrmacher. Simulation-supported engineering of self-adaptive software systems. In *2021 Winter Simulation Conference (WSC)*, pages 1–12, 2021. doi:10.1109/WSC52266.2021.9715324.

- A. Ruscheinski, **P. Wilsdorf**, J. Zimmermann, U. van Rienen, and A. M. Uhrmacher. An artefact-based workflow for finite element simulation studies. *Simulation Modelling Practice and Theory*, 116:102464, 2022. doi:10.1016/j.simpat.2021.102464.

- A. Ruscheinski, A. Wolpers, P. Henning, **P. Wilsdorf**, and A. M. Uhrmacher. SIMPROV: Provenance capturing for simulation studies. In Preparation.

2 Background and Objective

Modeling and simulation (M&S) has been established as a scientific method to answer questions about systems, in which experiments with the real system are not feasible. Simulation studies are iterative processes that involve phases of model building and model refinement, interweaved with the design and conduction of various types of simulation experiments. Conducting simulation studies has been considered to be an art and a science [6]. This duality arises from the inherent complexity of the process, wherein, despite the presence of life cycle models, no fixed sequence of steps can be prescribed. Instead, M&S studies require comprehensive knowledge and experience about the various tasks involved in solving a particular problem, the computational methods to be employed as well as the application domain.

This is particularly crucial for simulation experiments. For instance, what simulation experiments are needed, how they are specified and executed, and how results are interpreted depends on various aspects, such as the current stage of the study and the M&S approach used. A variety of decisions have to be made, in choosing from the vast number of available experiment types, analysis methods, as well as available tools and libraries. All of these decisions may influence the outcome of individual simulation experiments, the conclusions that are drawn referring to the behavior of the simulation model, and their validity in answering specific questions about the system of interest [10].

This chapter provides an overview of the M&S life cycle as well as the research questions and steps involved (Section 2.1). Then, the types of simulation experiments that will appear in the case studies of the upcoming chapters are categorized regarding their purposes in the M&S life cycle, and possible methods to be used (Section 2.2). For the different types of simulation experiments, available tools are discussed (Section 2.3). Next, recent developments in computerized support for simulation experiments are outlined (Section 2.4). The chapter concludes with identifying the main objective of the dissertation, i.e., supporting simulation experiments by automatic reuse and adaption (Section 2.5), and a summary (Section 2.6).

2.1 The Modeling and Simulation Life Cycle

Simulation studies evolve through different phases [74, 2, 3], intertwining successive model building with complex simulation experiments used for calibration, validation, or analysis. Depending on the overall research question of the simulation study and the progress so far, different types of simulation experiments may be applied.

2.1.1 Research Questions of a Simulation Study

Simulation studies are always conducted with an overall *research question* in mind. This is underlined by the reporting guideline ODD [75], which starts by asking for "a concise and specific statement of the purpose(s) for which the model was developed". As purposes they name, e.g., prediction, explanation, description, theoretical exposition, illustration, analogy, and social learning.

According to Cellier research questions can be categorized into one of five goals, each accompanied by a definition and an example [15]. The validity of the developed model decreases from top to bottom:

1. *Design.* Simulation studies may be conducted with the goal to design or manipulate a system such that it behaves in a certain way. This works especially for applications that are already well understood so that the designed system will behave as predicted by the model. An example is the design of electrical circuits with specific behavior.

2. *Control.* When there is sufficient understanding but the system's behavior is influenced by various factors, simulation models can act as controllers for the system. This necessitates a validated model that integrates feedback from the system. For instance, simulation models may be used for process control in chemical industry.

3. *Analysis.* Many simulation models are built to identify the causal relations or structures that make up a system's characteristic behavior. This is the case if the model possesses still enough validity, however, cannot be trusted for design or control purposes. An example includes models that represent the fundamental motifs of a biological system and are used to analyze the interactions between their variables.

4. *Prediction.* If a model is deemed sufficiently accurate, i.e., it was validated against data from the real system, it may be used to make predictions about the future of the system. This is the case, e.g, for models of economic systems.

5. *Speculation.* There are many applications, in which the mechanisms of the real system are unknown. Thus, the process of modeling is in the focus. Different hypotheses are proposed and tested to find a valid simulation model that explains the phenomena observed in the real world. This is true, e.g., for the simulation of social systems.

Kleijnen et al. find a different categorization, in which they distinguish between the following three objectives [76]:

1. *Developing a basic understanding.* This includes substantiating or refuting hypotheses, and gaining new insights into the underlying mechanisms of the modeled system that are not well understood and where data is scarce. The model may also help analysts to identify research questions and scenarios to explore if these are not known beforehand.

2. *Finding robust decisions or policies.* These studies explicitly focus on finding a robust rather than an optimal solution. Due to the large number of parameters, events, and assumptions in the model, and uncertainty in its inputs, obtaining an optimal solution is not always feasible. Robust solutions can therefore be defined as "works well across a broad range of scenarios".

3. *Comparing decisions or policies.* Those simulation studies are used for making predictions; however, it is emphasized that no single prediction but a comparison of a number of scenarios and measures is required to select "good" policies or decisions.

Put in other words, the overall research question of a simulation study—like research questions in general—may be of exploratory nature [76, 77], confirmatory nature [77], or for generating predictions including optimal (or robust) solutions [76].

2.1.2 Steps of the Life Cycle

Variations of the M&S life cycle have been widely discussed. A survey by Timm and Lorig compares various proposed life cycles (e.g., Maria [74] and Banks [78]) regarding the included steps [79]. Figure 2.1 summarizes the relationship of the steps and products considered in these various life cycles (by extending figures by Sargent [3] and Balci [80]). The figure also subsumes the steps of additional life cycle models, such as [81, 2, 4].

According to Balci, VV&T (validation, verification, and testing) is a continuous activity throughout the life cycle to ensure a credible model, and therefore needs to be considered explicitly in the life cycle [80]. Also, the life cycle is not sequential but the transitions shown in the figure can be used back and forth to revise certain parts if necessary.

From Problem to Conceptual Model

The first step, when starting a new simulation study is to *define the problem*. For this, the problem, of course, first needs to be recognized and communicated by the decision makers, domain experts, etc. to the M&S experts [80]. They then need to *define the research questions* to be answered by the simulation model [38] and *abstract* from what has been observed in the real system to define the problem boundaries to be large enough to capture the essence of the problem [2].

The next step is *investigating the system* to identify what input variables, entities and causal relations shall be included in the model, what level of aggregation is suitable to be considered in the model, and how the model can be decomposed into submodels—all of this with respect to the research question [80]. This also may involve experimenting with the real system and collecting real system data [3], and selecting appropriate response variables [82]. In this process, also the sources of uncertainty need to be identified, and suitable input distributions for random variables have to be assumed [74]. Other aspects of preparing the study may include assessing different solutions, such as the feasibility of alternative M&S approaches [80], and planning project constraints, such as the time frame of the study [74]. Furthermore, *hypotheses* to be probed by the model have to be derived [3], and model requirements, both functional and non-functional, need to be specified [2].

All the information and data collected during the planning and investigation phase can be incorporated into *building the conceptual model*. The conceptual model may be represented by flowcharts and various other diagramming techniques or pseudocode [80]. Recent approaches, furthermore, see the conceptual model as collection of information and data about the system to model, as well as constraints, assumptions and simplifications [82, 83]. Accordingly, the conceptual model can be seen as "a repository of high-level conceptual constructs and knowledge specified in a variety of communicative forms (e.g., animation, audio, chart, diagram, drawing, equation, graph, image, text, and video) intended to assist in the design of any type of large-scale complex M&S application" [2]. In addition to these communicative forms, formal and semi-formal approaches for representing the conceptual model have been used [83]. Section 4.2 discusses advantages and disadvantages of different representations.

But before it can be used further, it is necessary to *validate the conceptual model* by checking that its logic is reasonable, complete, and consistent [84]. Validation techniques for the conceptual model have been suggested, and the primary validation techniques used are face validation and structured walkthroughs [3]. This may also include the validation of data, where the following questions need to be asked: "Does each input data model possess a sufficiently accurate representation?"; "Are the parameter values identified, measured, or estimated with sufficient accuracy?"; "How reliable are the instruments used for data collection and measurement?"; "Are all data transformations done accurately?"; "Is the dependence between the input variables, if any, represented by the input data model(s) with sufficient accuracy?"; and "Are all data up-to-date?" [80]. A careful collection and evaluation of data is especially important since data are used for various purposes in the simulation study, and as such become a crucial part of the conceptual model: as input for the simulation model, for calibration and validation [3].

From Conceptual Model to Experiment Designs

From a conceptual model, the simulation study continues by *implementing the simulation model* in a modeling language of choice. This may be a general purpose programming language or a

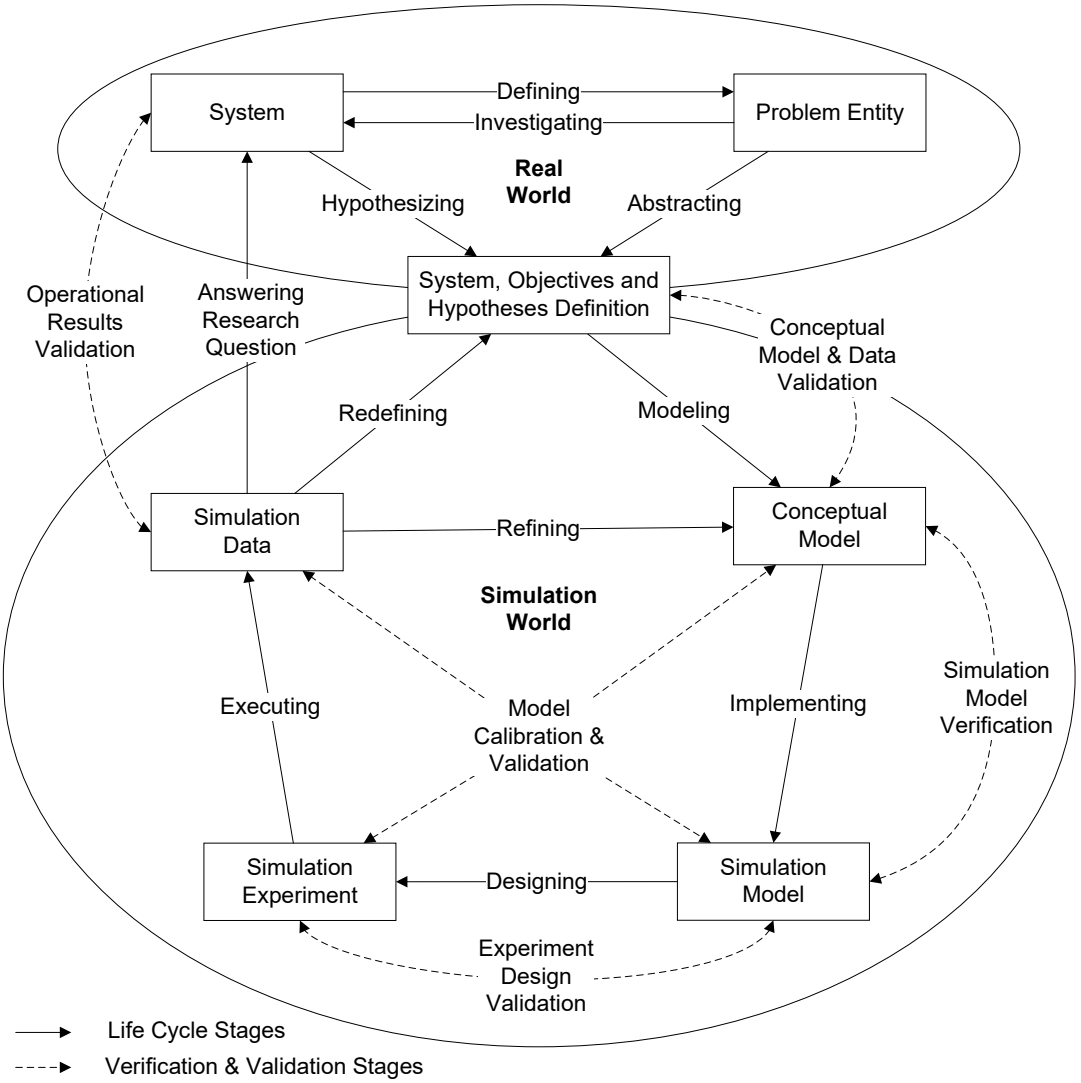


Figure 2.1: The modeling and simulation life cycle, adapted from Balci [80] and Sargent [3].

domain-specific modeling language. After the model has been programmed, *verification of the model* is conducted to ensure that the model was implemented correctly with respect to the conceptual model. Here, in particular, it needs to be checked that simulations are error-free, and implemented functions behave as expected, such as random number generators and program logic [3]. For verification, two approaches exist: static and dynamic analysis. Static analysis is concerned with looking at the model code, e.g., proofs of correctness, whereas dynamic analysis executes the model and runs tests on the generated output traces.

With the executable simulation model, one can then conduct simulation experiments by first *designing* and then *executing* it. Various things have to be considered during the design of experiments: e.g., are the simulation algorithms correct, have appropriate statistical methods been used, and is the number of replications sufficient [80]. Thus, *validation of the experiment design* is a crucial step.

Simulation experiments can be used for *calibration of the model*, i.e., fitting parameter values such that some real-world data are reproduced or requirements from the conceptual model are satisfied. Another important task in the M&S life cycle is *validation of the model*. Techniques for validation include comparing with data from the real system, checking (formally) defined behavioral requirements by hypothesis testing, comparing to other (valid) models, or performing face validation (i.e., consulting domain experts whether the model output is reasonable, usually

by providing them with a graphical representation of the results) [3, 85]. Besides validation and calibration, the various types of experiments can serve other purposes, e.g., allowing analysis of the model and providing insight into the real system (see Section 2.2.1).

An important preparatory step of experimental design is surrogate modeling [86]. Surrogates, also known as response surfaces, metamodels, emulators, or auxiliary models, are built whenever the execution time per simulation run needs to be reduced. Additionally, surrogate models are an integral part of some experiment types, such as sensitivity analysis (see Section 2.2.3). They are validated just like the simulation model and thus can have an own life cycle [81]. Once validated, the surrogate can be used instead of the simulation model to conduct analyses as it is usually faster and easier to use [87].

From Simulation Results to Conclusions about the System

When the simulation experiment has finished, the simulation data have to be analyzed, visualized and interpreted by (statistical) analysis of the runs, and it needs to be decided if additional experiments are necessary [4].

Furthermore, if the model could not be successfully calibrated or validated, it has to be decided if and what kind of changes of the simulation model are needed. Consequently, rarely only one iteration of the described M&S life cycle is required, especially if a valid model needs to be built for understanding. Thus, *refinements* of the conceptual model and subsequently the simulation model are necessary, creating several versions of the simulation model (and the simulation experiments as well) [80]. Moreover, the research questions may need to be *redefined* based on the outcome of the simulation experiment.

Finally, once the model yields satisfying results, one can *answer the research question* and draw conclusions with respect to the real system. This includes communicating the results to decision makers, and gaining new insight into a process, depending on the research question of the simulation study (see Section 2.1.1). If the model is to be used for predictions and decision making, *operational validity* may need to be established to ensure that the model's range of behavior is appropriate for the intended purpose, i.e., additional checks about the model's behavior are required [3].

2.1.3 Specifics of Modeling and Simulation Approaches

The suitability of a M&S approach depends on the research question of the simulation study and the system of interest. For example, if the number of reacting molecules in a cell biological system is small, a stochastic modeling approach may present a more suitable alternative compared to deterministic approaches [88, 89]. Additionally, if the spatial distribution of molecules controls how molecules interact, a spatially resolved approach may be required [90, 91].

However, even if a stochastic or spatial approach has been decided upon, this does not answer the question of which of the many available methods to select. This applies to algorithms for interpreting and executing a stochastic or spatial model [92, 93], to the languages in which models are described [94, 40], and to the analysis of simulation results [95]. Similar decisions have to be made when conducting simulation studies in the social sciences (e.g., whether to use an agent-based modeling metaphor, or whether to run the model using discrete step-wise or discrete-event based execution semantics), finite element analysis (FEA) in electromagnetics (e.g., whether to use adaptive mesh refinement), and virtual prototyping of heterogeneous systems (e.g., whether to use SystemC, Verilog, or Modelica as a modeling language).

Referring to the case studies conducted in this dissertation, some peculiarities in the life cycles of the corresponding M&S approaches can be noted. For example, models of cell biological systems are often developed for understanding the underlying dynamics. Thus, there is a strong focus on successive calibration and validation of the model structure and associated parameters. To that end, the simulation study can be data-driven, integrating lab results obtained at multiple

organizational scales [96], and thereby assigning modeling and experimenting activities based on data a central role in the life cycle.

In virtual prototyping studies of heterogeneous systems, the use of data is less pronounced, and the focus is naturally on the design and analysis of systems based on requirements and constraints defined at the beginning of the study. Thus, optimization based on constraints as well as verification of system properties play a vital role for these studies. Moreover, the process of making the system requirements explicit may need to be made more explicit.

For FEA, a specialized workflow could be identified [52]. Therein, the simulation model is divided in a physical model and a geometric model, and different information needs to be collected during conceptual modeling (e.g., CAD images), which are then processed during model building. Moreover, additional kinds of simulation experiments (i.e., error and convergence tests) are required, which are inherent to the M&S approach based on partial differential equations and discretization of the domain of interest into smaller, finite-sized elements, forming a mesh. These specifics of the life cycle need to be considered in conducting as well as documenting these studies [62].

For developing realistic agent-based models, social simulation studies rely on the collection and assessment of empirical data. Those may include data from psychological experiments or demographic data from a variety of sources [69]. Consequently, stages of primary and secondary data collection have to be taken into account in the life cycle. In applications where data are sparse, methods are sought that allow validating without data from the real system [97]. Furthermore, social simulation studies are driven by various behavioral theories to be explored and compared [73]—another aspect to be incorporated in the model building process. The exploration of these theories via simulation experiments, and the results and insights gained from those, will also allow finding new research questions to be explored in the next iteration of the M&S life cycle.

Looking beyond the simulation studies used in this dissertation, further activities in the M&S life cycle occur, e.g., when developing and using digital twins as this includes the collection and use of "real-time sensor data, historical data, etc. to mirror the life of its corresponding twin" for making predictions or optimizing processes [98]. Furthermore, in symbiotic simulation, the simulation and a complex adaptive system interact in a mutually beneficial manner, and what-if analyses are crucial for weighting up suitable options for controlling the physical system [99]. Those "closed loop" activities of perceiving information from the real system and controlling the system based on the simulation results would require special attention in the M&S life cycle.

In addition to these differences referring to the stages in the M&S life cycle, there are also specifics in the types of simulation experiments executed, which are discussed in the following section. Moreover, the structure of a simulation experiment can differ for the various kinds of simulation approaches. Different configuration parameters may exist for the different simulation algorithms and analysis methods, which will be investigated further in Chapter 3. Furthermore, for actually specifying a simulation experiment and for automatically generating it, domain-specific context knowledge is required, see Chapters 4, 5.

2.2 Simulation Experiments in the Modeling and Simulation Life Cycle

In the discussion of the life cycle shown in Figure 2.1, the simulation experiments had various roles (i.e., calibration, validation, and analysis), which will be examined in more detail in the following. Thereafter, the general structure of a simulation experiment is presented and, subsequently, an overview of experiment types and corresponding experiment design methodology will be given.

2.2.1 Roles of Simulation Experiments

Simulation experiments can fulfill different roles depending on the current stage of the life cycle, the research question of the simulation study, and the requirements that have been defined during the planning and conceptualization stages. Thus, simulation experiments may at first be categorized according to their role in the assessment of the model during or after model building.

Simulation experiments conducted during the M&S life cycle can be used for model verification, validation, calibration, and analysis. Whereas calibration, validation and verification occur mostly during the model building and refinement process, analysis experiments are also conducted after model building with the produced model.

Generally, *calibration* or *fitting* "applies mathematical and statistical techniques to a set of simulation I/O data, to (i) estimate the [simulation model's] parameter values, and (ii) evaluate the estimated parameter values with respect to the data set, using quantitative criteria." On the other hand, verification asks the question "did I build the model right?", while validation is concerned with "did I build the right model?" [2]. More precisely, "[m]odel *verification* is defined as 'ensuring that the computer program of the computerized model and its implementation are correct'" [3]. A related topic is model credibility, which is concerned with "developing in (potential) users the confidence they require in order to use a model and in the information derived from that model" [3]. In contrast, *validation* is the "substantiation that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application [and purpose [76]] of the model" [3]. Several versions of a model are usually developed prior to obtaining a satisfactory valid model. Thus, developing a simulation model is an iterative process with successive calibration, validation, and verification, e.g., via regression testing [100, 101].

Analysis experiments encompass a wide range of investigations conducted throughout the simulation study. They include sensitivity analysis, parameter estimation, simulation optimization, and other types of experiments aimed at understanding system behavior, identifying influential model parameters, optimizing system performance, or comparing different policies or strategies, as well as toy duck analysis of individual time series.

The role of a simulation experiment can also be categorized according to the purpose of the analysis. Balci has identified (1) comparison of different operating policies, (2) evaluation of system behavior, (3) sensitivity analysis, (4) forecasting, (5) optimization, and (6) determination of functional relations [80]. Barton classifies the more fine-granular (sub)goals of a simulation study into (1) early goals (i.e., taking place in early iterations of the life cycle): validation, screening variables, (2) middle goals: sensitivity analysis, understanding, predicting, selecting the best configurations, (3) late goals: optimization and robustness of the parameter settings [102]. The late goals can be supplemented by uncertainty and sensitivity analysis as part of robustness analysis [103]. Once a satisfactory parameter configuration could be found, sensitivity and uncertainty analysis should be conducted to reinforce the result and to test the robustness of the solution, and quantify the uncertainty.

In simulation studies that are conducted for understanding (see, e.g., the case studies in Section 1.2) the steps are closely intertwined and carried out iteratively until a model with satisfactory accuracy has been developed [100]. In these studies, the question of early, middle, or late goals is less important, and the broader categories appear more suitable. This dissertation, thus, will focus on validation, calibration, and analysis as roles for simulation experiments (as has been done, e.g., by [43]). Verification can mostly be neglected, since software verification is often done based on static methods that do not require simulation.

2.2.2 Structure of a Simulation Experiment

Conducting a simulation experiment involves several tasks. Figure 2.2 shows the layered view of a simulation experiment with a stochastic model that varies model parameters as proposed by Rybacki et al. [16]. The specification of a simulation experiment is situated at the topmost layer.

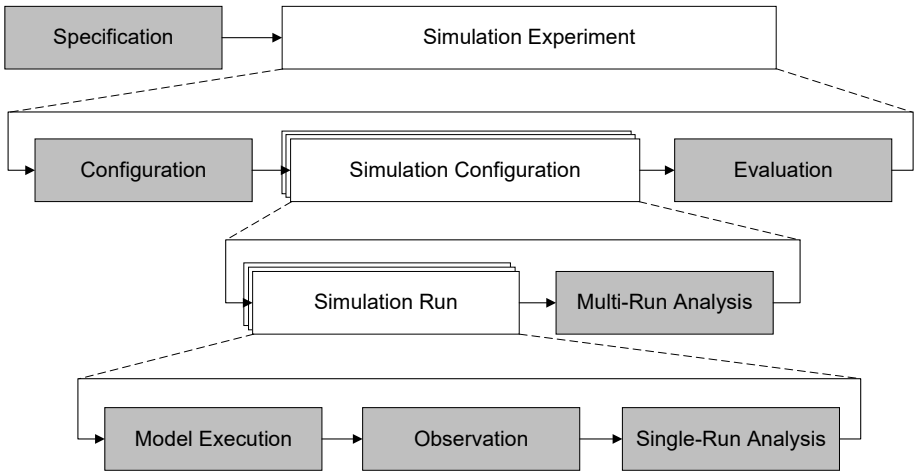


Figure 2.2: Structure of a simulation experiment that varies the model parameters, adapted from Rybacki et al. [16]. Experiment specifications and subparts of the specifications shown in grey.

Its purpose is the description of all information required to run the simulation experiment and to complete all the tasks at the lower layers. The second layer of a simulation experiment focuses on the configuration, execution, and evaluation of the simulation model under different parameter settings. Therein, the simulation model is treated as a black box by the simulation experiment [86]. During the evaluation task, feedback about the different parameter configurations is provided, so that additional parameter configurations can be created and evaluated if necessary—depending on the type of experiment. After evaluating all parameter configurations, the final result of the overall experiment is returned. In case of a stochastic simulation model, multiple replications have to be executed and analyzed, typically by calculating averages for each configuration (depicted at the third layer). In each replication, the simulation model is executed, model variables are observed, and the results of the single run are analyzed (depicted at the bottom layer). The feedback loops at the lower two layers allow adjusting the number of replications (e.g., via construction of confidence intervals) as well as the length of the simulation run (e.g., by steady-state analysis) depending on the obtained results [38]. If the simulation is deterministic, only one simulation run has to be executed, and thus layer three can be omitted.

For a single simulation run or across those single runs, the decisions made during the simulation experiment include whether to perform a terminating or steady-state simulation, estimating the distributions of stochastic model components, selecting the initial conditions or the duration of the warm-up period, choosing the final conditions such as simulation stop time or number of events completed, choosing variables to observe as well as the observation time step, and deciding on an appropriate balance between run length and the number of replications [104]. Further issues include choosing a method for random number generation, and deciding whether to use an appropriate variance reduction technique. They are, however, not always covered by the experiment specification but subject of pre- or postprocessing on the runs.

Besides the structure shown in Figure 2.2, some experiment types, such as statistical model checking or steady state analysis, do not rely on varying model configurations. In addition, the experiments may not only change the parameters but also the model structures from one configuration to the other. Also, single stochastic simulation runs may be treated as a simulation experiment. For instance, with agent-based models, single trajectories can reveal important insights about the behavior and interactions of specific individuals, where mean trajectories would return biased results. In addition, this general structure of a simulation experiment does not only relate to time course simulation but also, e.g., to simulations in the frequency domain as often done in FEA.

2.2.3 Types of Simulation Experiments

During a simulation study, a variety of simulation experiments is conducted, be it for calibration, validation, sensitivity and uncertainty analysis, what-if exploration, or optimization of the system [104].

In the following, important experiment types are introduced that are also used as case studies or for discussion and illustration in this dissertation. These include steady-state estimation, parameter scans, sensitivity analysis, parameter estimation, optimization, statistical model checking, what-if analysis, and error and convergence control.

Note that there may be further kinds of simulation experiments that are concerned, e.g., not with the behavior of the model but with the performance of a simulation. Those may be covered in extensions of this work. E.g., performance requirements may be explicitly specified and then checked during model validation, or specific performance benchmarks can be run for the finally developed model.

Steady State Estimation

Stochastic time course simulations can be categorized into terminating and non-terminating simulations [38]. Terminating simulations run for a finite time horizon until a specific stop condition is reached (e.g., a stop time, a state, or when a specific event occurs). In contrast, in a non-terminating simulation, the simulation runs continuously or at least over a long period of time, allowing one to study the long-term behavior of systems. This includes simulations that, after an initial transient phase, reach a steady-state or a cycle (i.e., they oscillate). In both cases, the "end of the simulation", namely when the steady state or cycle has been reached, is not fixed and thus has to be detected. The detection of oscillations in stochastic simulations is a complex issue and requires specialized methods (such as [105, 106]). The following therefore will solely discuss methods for steady state detection.

Figure 2.3 illustrates the probability densities of the stochastic process given by the stochastic simulation replications at time points $i_1, i_2, \ldots$ The figure shows that there is more variance at the beginning of the simulation. These are known as the transient states. After this initial transient phase, the simulation reaches a steady state, i.e., the densities do not vary much for the rest of the simulation and the expected value $E(Y_i)$ converges to the steady state mean $E(Y)$.

Steady state analysis is applied for a number of problems concerned with correctly starting and stopping stochastic simulation experiments to achieve accurate results. This includes cutting off the initial transient (known as warm-up phase), steady state detection, and determining the minimum number of replications [105]. Furthermore, steady state analysis is closely related to stability analysis, which is concerned with the stability of a (steady) state under arbitrarily small parameter perturbations [107]. Steady states, if they are attracting all nearby initial conditions, are said to be stable, and unstable if they repel some initial conditions. Bifurcation theory is the study of critical points in the parameter space where small perturbations result in changes in the steady state behavior of a nonlinear system [108, 109].

Steady state detection can be conducted in the early phases of a simulation study as a preprocessing step to determine the time horizon to be simulated in the simulation experiments to follow. Furthermore, steady-state estimation may be conducted during a simulation experiment (such as a parameter scan or sensitivity analysis) to analyze the individual stochastic simulation runs as well as the various configurations of the experiment design, see single- and multi-run analysis blocks shown in Figure 2.2.

But estimating the steady state behavior can also be the primary goal of the simulation study. In these studies, one is not interested in the behavior over time, but rather the expected value or the distribution of results at steady state, e.g., the "normal" operational behavior of a factory [38]. Furthermore, studies might be designed to explicitly identify bi-stable states in a cell signaling network [107].

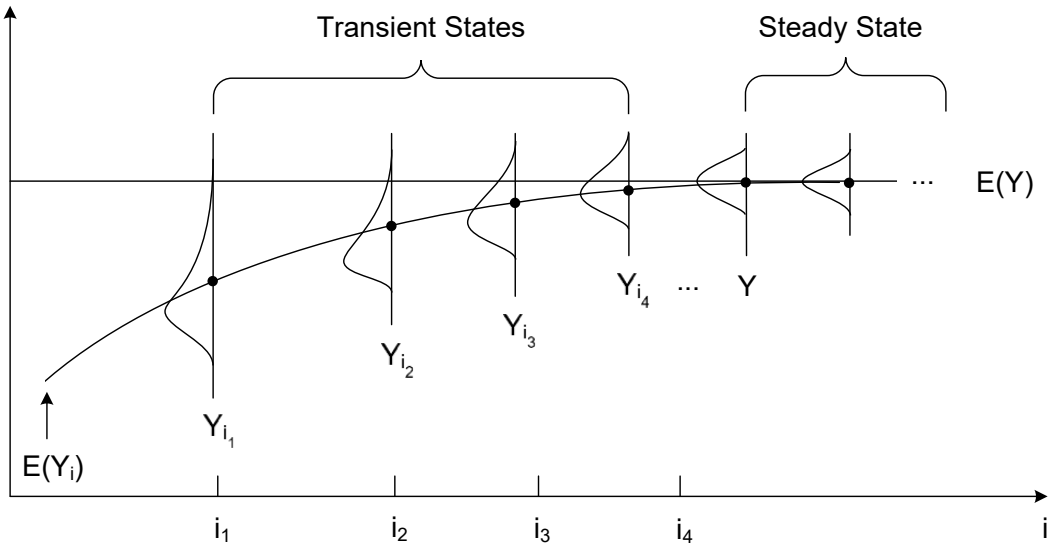


Figure 2.3: Transient and steady state behavior of a simulation given by the underlying stochastic process with the random variables $Y_{i1}, Y_{i2}, \ldots$, increasing time indices $i_1, i_2, \ldots$, and a specific set of initial conditions; adapted from Law and Kelton [38, Chapter 9].

Available methods for detecting the steady state are plentiful, which underlines the difficulty of selecting a suitable method for a given problem [110]. Some methods use multiple (shorter) simulation runs to detect the steady state and cut off the warm-up period. For instance, the method of Welch computes a moving average over the replications and thereby smooths the simulation output data. The length of the warm-up period is chosen by visually inspecting where the mean seems to converge. The mean and confidence interval of the steady state variable can then be estimated by only including simulation output beyond the warm-up period that was determined using Welch's method (known as the replication/deletion approach) [38]. Other methods for determining the warm-up period include bias detection tests and marginal standard error rules [111]. In contrast, other methods operate based on one long simulation run. The batch means method, for instance, partitions the output of one long simulation run into a sequence of disjoint batches and computes their means. Since the batch means can be considered independent samples, they can be used to get a confidence interval of the estimator. The length of the simulation run is sequentially increased until an acceptable confidence interval can be constructed [112]. Various algorithms exist for the batch means method [113]. The moving windows approach also works on one long simulation run but moves a window of fixed size through the time series and updates the mean according to the values inside the window [114]. If the standard deviation falls below a given threshold, the end of the warm-up phase has been detected.

Parameter Scan

Parameter scans (also referred to as parameter screening) can be viewed as the simplest form of sensitivity analysis according to Cacuci et al [115]. However, in contrast to other sensitivity analysis methods, their main purpose is to get a first impression of the model's behavior. They are, therefore, used for exploration and sometimes ranking of parameters rather than quantifying how the uncertainties in the model inputs $(X_1, X_2, \ldots, X_k)$ affect the model's response Y [116]. Consequently, parameter scans rely on relatively simple experiment designs and simple analyses, e.g, visual interpretation of the results. For a detailed discussion of sensitivity and uncertainty analysis see the following subsection.

Parameter scans are a good choice if no research question has been defined yet and the model

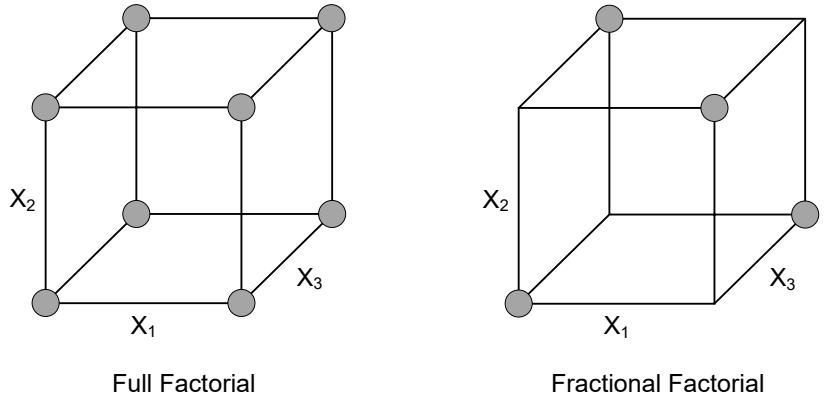


Figure 2.4: 2^k full factorial design (left) and 2^{k-1} fractional factorial design (right), adapted from Sanchez [120].

needs to be explored to find interesting input-output relationships. Also in the case where a research question already exists, parameter scans are essential in the early phases of a simulation study for exploring alternative implementations and their emergent behaviors in a "wind it up and let it run" approach. This is of interest not only when models are built from scratch but also when composing or extending models and then iteratively refining them by adding new parameters, entities, or rules. Another important task that can be tackled by using parameter scans in the initial phase of a simulation is to determine which of the many model parameters are really important. Screening, thus, allows reducing the number of parameters to be explored further in the next phase to the most important ones, as including all of them would be infeasible and even unnecessary. For instance, during a simulation study of the COVID-19 pandemic, the parameter space could be successfully reduced from over 900 parameters to 60 parameters that were included in the subsequent uncertainty quantification [117]. Parameter scans may also be used for validation and verification, however, more suitable experiment types exist [118]. Moreover, parameter scans are required as integral part of other experiment types, such as global sensitivity analysis, selecting the best model (e.g., by ranking and selection [119]), building a surrogate model, or testing hypotheses.

To conduct a parameter scan, first, the independent variables of the model have to be identified. Those are the parameters that can be varied during the parameter scan to observe the outcome of one or more dependent variables (model outputs) [102]. The decision about which parameters to include in the experiment is crucial, as one might miss important effects and interactions due to a prior bias to whether some parameters are important or not. The simplest and easiest way to realize a parameter scan are full factorial experiment designs, in which all combinations of parameter values are explored. They provide a fine-gridded view of the entire parameter space, and allow constructing accurate response surfaces. However, these designs suffer from state-space explosion, and therefore are unfeasible for realistic models.

A carefully chosen experiment design can alleviate the trade-off between computational performance and information gain. Experimental designs indicate how to vary the settings (levels) of factors to see how different configurations affect the response [120, 86]. A factor mostly refers to parameters but may also represent the structures of a model, and the levels of a factor are a discretization of the value range or an enumeration of the structural features to be explored. A particular combination of factor levels is called a design point. Figure 2.4 (left-hand side) illustrates the full factorial case with $k = 3$ factors and $m = 2$ levels, in which m^k combinations have to be tested. The fractional factorial case (Figure 2.4, right-hand side) requires only m^{k-p} combinations but at the cost of coarser granularity, with p characterizing the fraction of interest.

Kleijnen et al. recommend different (fractional) designs depending on the number of factors and the complexity of the response surface [76]. A noteworthy class of efficient fractional factorial

designs are Latin hypercube designs as they are capable of capturing complex, non-smooth response surfaces. The number of design points in a Latin hypercube grows linearly with the number of factors, k, instead of exponentially. Moreover, Latin hypercubes have several advantageous properties that can be achieved if the design is constructed with the appropriate algorithms. In particular, a Latin hypercube is orthogonal if there are no pairwise correlations between any two factors in the design. This guarantees that each main effect and interaction effect of a factor can be estimated independently of the effect of any other factor or interaction in the model, thereby simplifying the analysis [121, 122]. In addition, Latin hypercubes allow for good space filling, i.e., an even distribution of design points across the parameter space. Nearly-orthogonal Latin hypercubes (NOLH) strive to balance these two properties [123].

Sensitivity Analysis

Sensitivity analysis (SA) investigates how the variation in the output of a model can be attributed to variations of its input factors [124, 125]. The general formulation of the SA problem is given as

$$Y = g(X) = g(X_1, X_2, \ldots, X_n)$$

where Y is the model output, $X = (X_1, X_2, \ldots, X_n)$ is the vector of input factors, which belong to the input variability space with dimensionality n, and g is the function that maps the input factors to the output, called the response function. For realistic simulation models, this relationship between input and output can hardly ever be determined analytically. Therefore, it is approximated and quantified by using suitable experiment designs and analyses that measure quantitative sensitivity indices.

Figure 2.5 provides an overview of the process involved in conducting a (global) SA, offering a bird's eye view of the entire procedure. The variability (also known as uncertainty) of the model output can stem from different sources, which include the model parameters, model structure, resolution levels, and the input data used as basis for the model [126]. During a SA experiment, a subset of these sources is analyzed.

Conducting a SA experiment typically encompasses four steps [125]. The first step involves setting up the experiment, which includes selecting the input factors and model outputs, as well as choosing the appropriate SA method to address the specific research questions. The second step involves input sampling, which refers to the experiment design. The third step is model evaluation, wherein the model is executed using the input samples, and the output metrics are observed. Finally, the post-processing step involves calculating sensitivity indices based on the overall model uncertainty and visualizing the results. The analysis results obtained provide valuable feedback on both the model and the input data. It is important to note that the experiment design is often intertwined with the choice of the specific analysis method, eliminating the need for separate explicit selection of these steps.

SA provides a wealth of valuable insights, as highlighted in previous research [127, 128]. Factor screening aims to identify important factors within a SA, as described in the previous section on parameter scans. Factor ranking and prioritization aim to quantify the effects of input factors, and can be used to guide the model simplification, i.e., reducing the number of factors, model entities, or level of abstraction to those that have the greatest influence on the model's behavior. Variance cutting involves reducing the output variance to a defined tolerance level, ultimately resulting in a more robust model. Robustness analysis evaluates the robustness against small perturbations, such as examining the effect of small changes in temperature on the optimal battery runtime of a microprocessor, or the effect of small environmental changes on the stability of an ecosystem. Related to this is analyzing the direction of change to determine whether a particular change in the input yields a positive or negative effect on the model's output. In addition, factor mapping aims at risk assessment by identifying specific input conditions that lead to critical output values.

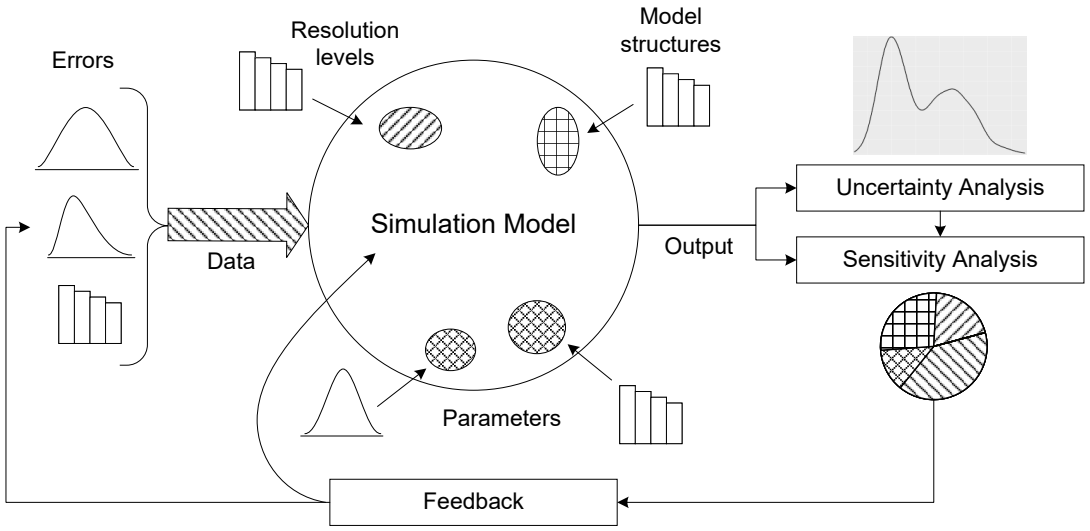


Figure 2.5: Overview of a (global) sensitivity analysis, adapted from Saltelli et al. [126]. Input data, resolution levels, model structures and parameters contribute to the model's input uncertainty. Uncertainty from these heterogeneous sources is propagated through the model to derive a distribution of the model output (uncertainty analysis). This output uncertainty is then decomposed according to its sources (sensitivity analysis). The result provides feedback about the model and its inputs, guiding future model adjustments and the collection of data.

In addition to these insights, SA is closely related to various other M&S tasks, including uncertainty quantification, calibration, and validation of simulation models, as discussed in the review by Pianosi et al. [125]. Furthermore, SA can drive the iterative development of simulation models, guiding decisions on whether to split a parameter into multiple more fine-granular parameters, introduce additional species to the model, or make other adjustments [129].

Due to this variety of application settings, choosing an appropriate SA method is crucial since not all methods are suitable for the question at hand [125]. Methods can be categorized into local and global SA methods [124, 125]. For local SA, the focus is on analyzing the model's behavior around a specific point in the input space. Those methods thus are suitable for robustness analysis, and analyzing direction of change. This is achieved by considering a base case for the model inputs, denoted as $\boldsymbol{x}^0$. Those reference values are then varied one-at-a-time (OAT), assigning and evaluating a sensitivity case $\boldsymbol{x}^+$ for each of the parameters of interest individually. When model parts are changed, the analysis resembles a what-if analysis involving the exchange of rules or entire submodels [130]. Local SA, however, cannot capture the relationships between input factors. Moreover, it depends heavily on the chosen reference point. Multistart perturbation-based methods, such as the Elementary Effects Method of Morris [131], try to alleviate this problem. However, for understanding the global behavior of the model, a global SA method should be used. Those methods calculate a suitable measure of the global response using an all-at-a-time sampling approach on the entire input space, e.g., based on a Latin hypercube design. Methods for global SA can be correlation- and regression-based, e.g., using the Spearman rank correlation coefficient (SRCC) or the partial rank correlation coefficient (PRCC) [125]. Other methods rely on variance decomposition to calculate the main and interaction effects between factors, such as the Sobol method [132] or the Fourier Amplitude Sensitivity Test (FAST) [133]. Methods can also be density-based, i.e., quantifying variations in the probability density function or cumulative distribution functions of the model output [125]. Furthermore, specialized methods exist for the various purposes of SA, such as the Classification And Regression Trees (CART) for factor mapping [125].

Parameter Estimation

In uncertainty quantification (UQ), both the forward problem ("characterizing the model outputs") and inverse problem ("learning about the model inputs") play crucial roles in understanding and managing uncertainties in simulation models [134]. The forward problem in UQ aims to characterize the model outputs based on known or assumed input parameter distributions. It involves propagating the uncertainties associated with input parameters through the model to assess the resulting uncertainty in the model predictions. Parameters are therefore sampled according to their input distributions, the simulation model is executed with these samples, and sample statistics of the model output are calculated. Thus, forward UQ is a central part of sensitivity analysis. In addition, Scheidegger et al. propose forward UQ as the final step in the M&S process [135].

On the other hand, the inverse problem in UQ seeks to determine the parameter values or model representations that best explain observed data or desired behavior. Within a M&S study, the inverse problem is also referred to as parameter estimation. Simulation experiments for parameter estimation can play several roles in the M&S process. Primarily, they are employed for the calibration of simulation models, aiming to determine the values of model parameters that best align the model's output with observed data [136]. This fine-tuning of parameters ensures that the simulation model accurately represents the real-world system it seeks to reproduce. For instance, while some kinetic reaction constants of a biochemical reaction network can be readily derived from literature, there are other components of the model that are not yet well-understood, and the true values of corresponding parameters remain unknown, requiring fitting [137]. In addition, simulation experiments for parameter estimation are not limited to parameter fitting alone. They also serve a vital role in uncertainty quantification. Once the final simulation model has been developed and calibrated, it is important to assess the uncertainty associated with the estimated parameter values and the resulting model predictions, which can be expressed by the posterior distributions. Furthermore, parameter estimation methods may be applied for model checking or design purposes [138].

A commonly used approach for parameter estimation is moment estimation, which involves estimating the parameters by calculating sample moments such as the mean and variance. Maximum Likelihood Estimation (MLE), a specific type of moment estimation, aims to find the parameter values that maximize the probability of observing the given data. However, MLE poses certain challenges. One of the drawbacks is its computational complexity. Additionally, there are situations where it is not possible to define a likelihood function. Another limitation is that MLE, following a frequentist view, does not account for parameter uncertainty or incorporate prior knowledge.

In contrast, Bayesian estimators treat unknown parameters, denoted by θ, as random variables. By adopting a Bayesian framework, the uncertainty associated with these parameters can be quantified using a posterior distribution, denoted by $P(\theta|D)$, given by Bayes' theorem:

$$P(\theta|D) = \frac{P(\theta)P(D|\theta)}{P(D)},$$

where D represents the observed data set, typically given by a trajectory with n time points $D = (D_1, \ldots, D_n)$. The posterior distribution incorporates both the observed data and any prior beliefs or information, given by the prior distributions $P(\theta)$ and $P(D)$, and the likelihood $P(D|\theta)$.

The likelihood function, however, is mostly intractable or too computationally expensive to evaluate directly. Instead, likelihood-free methods like Approximate Bayesian Computation (ABC) can approximate the likelihood via simulation runs, enabling the estimation of parameters and their associated uncertainties [140, 141, 142].

Popular ABC methods are rejection-based ABC, and the sequential Monte Carlo method. In both methods, first, from $P(D|\theta)$ a synthetic dataset Y is simulated for a parameter value θ, and

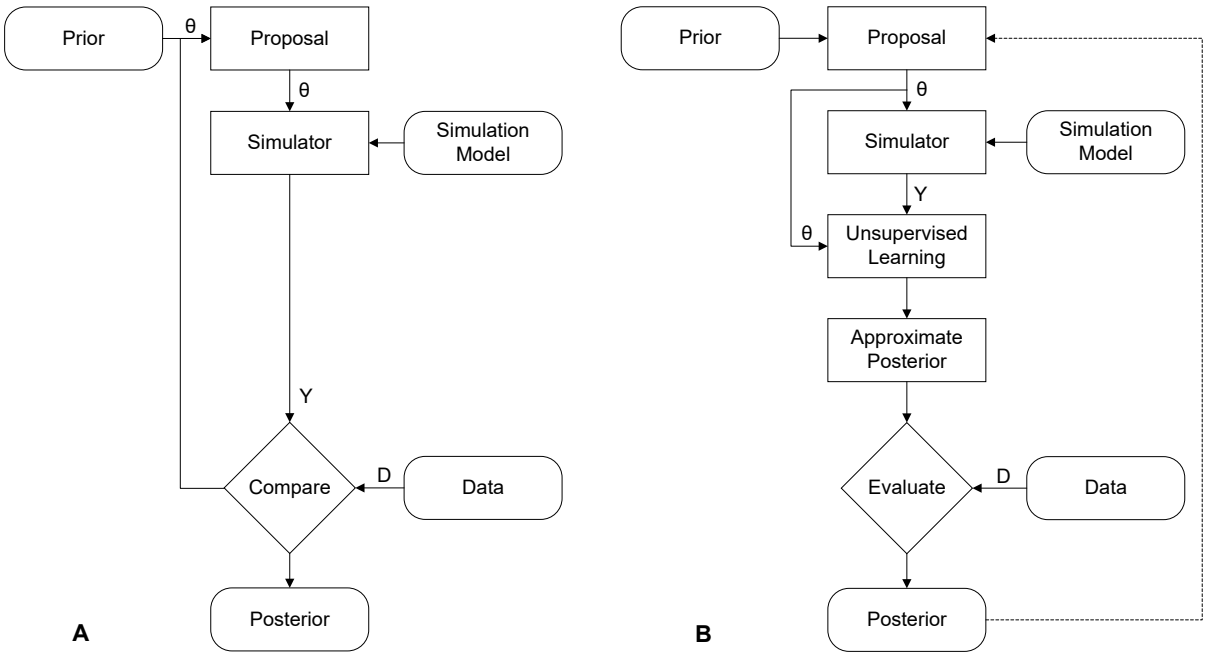


Figure 2.6: Two procedures for parameter estimation. The posterior distribution of a model parameter is estimated using A) Sequential Monte Carlo Approximate Bayesian Computation, and B) Simulation-based Inference based on unsupervised machine learning. Adapted from Cranmer et al. [139, Figures 1A and 1F].

a measure of closeness between simulated data Y and D is calculated [143]. As with the other experiment types, the simulated data may be produced using the simulation model directly or one may first build a surrogate model to produce the outputs efficiently. Based on a discrepancy measure, ABC accepts the parameter value of θ when the discrepancy is less than a pre-specified threshold value ϵ. As distance measure between observed data D and simulated trajectory Y of length n, root mean squared error (RMSE) or mean absolute error (MAE) may be used [144]:

$$\mathrm{RMSE}(D, Y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(D_i - Y_i)^2},$$

$$\mathrm{MAE}(D, Y) = \frac{1}{n}\sum_{i=1}^{n}|D_i - Y_i|.$$

The rejection sampling is repeated until enough parameter values have been accepted. Those are used to approximate the posterior distribution of θ.

The choice of ϵ is critical in the rejection ABC algorithm, as it determines the trade-off between accuracy of the approximation and computational efficiency. Particularly for small ϵ, lots of samples are required, as nearly all samples of θ are rejected.

As an improvement, a sequential variant of this procedure was developed, which is shown in Figure 2.6A. It starts by choosing a value of ϵ that ensures a sufficiently high acceptance rate. Then, instead of running many samples ahead, the sequential Monte Carlo approach successively generates more batches of samples, compares with the real data, updates the posterior, and repeats with smaller ϵ, morphing the prior into the posterior. As the posterior approaches the true probability distribution, with each iteration more and more samples are drawn from the region of interest, and the acceptance rate increases. The procedure therefore requires fewer samples overall. However, note that still millions of runs may be required to yield satisfactory results even for small numbers of parameters [142].

New methods, known as simulation-based inference (SBI), have emerged to address the limitations of ABC methods [139, 145]. They leverage state-of-the-art machine learning techniques, such as neural networks and active learning. Figure 2.6B illustrates the Sequential Neural Posterior Estimation (SNPE). Parameters are sampled from the prior distribution, followed by generating simulation data from these parameters. A deep density estimation neural network is then employed to learn the probabilistic association between the simulated data (or its features) and the input parameters. During the inference step, this trained neural network then acts as a surrogate model and is applied to empirical data D to determine the parameter space consistent with both the observed data and the prior, resulting in the posterior distribution. Parameter values that align with both the data and the prior are assigned high posterior probability, while inconsistent parameters receive lower probability. If necessary, an initial estimate of the posterior distribution can be used to guide the execution of additional simulations, ensuring the generation of data-consistent results. This adaptive approach is referred to as active learning [145].

In summary, simulation-based inference methods offer significant improvements over traditional ABC methods, such as reduced sample size, enhanced inference quality without the need for error thresholds, and elimination of redundant computational steps [139]. In particular, once the neural network surrogate model has been trained, it can be readily applied to new data. Thus, model calibration with respect to multiple data sets can be conducted efficiently.

Optimization

Simulation-based optimization aims to find the optimal or near-optimal combination of input factors for a simulation model [86, 146]. It aims to minimize or maximize a target objective, which is a function of the simulation output. To achieve this, an optimization algorithm orchestrates a sequence of candidate configurations, gradually converging towards a model configuration that provides an optimal or near-optimal solution. The goal is to reach this optimal solution by simulating only a small fraction of the total configurations that would be required by exhaustive fine-gridded parameter scanning [146].

The simulation optimization problem can be generalized as follows:

$$\min_{X \in \Theta} f(X)$$

where X represents the vector of model parameters, Θ defines the feasible region for the input parameters, and f is the objective function that summarizes the output variables into a single target variable [147, 148].

The optimization process, illustrated in Figure 2.7, follows an iterative loop. The optimizer generates and executes new simulation model configurations until a stopping criterion is met. This criterion could be reaching a predefined threshold for the value of the objective function or reaching a maximum number of iterations.

The demand for optimization techniques in simulation has grown significantly, leading to the development of numerous commercial and free software packages, as well as an expanding literature dedicated to the topic [148]. This widespread interest is also reflected in various applications of simulation-based optimization. Particularly, it has been utilized in engineering disciplines to discover optimal designs when building complex systems. For example, in the field of high-performance building design, which encompasses low-energy buildings and passive houses [149], simulation-based optimization techniques are employed to uncover optimal designs that minimize, e.g., energy consumption, CO_2 emission, and initial investment cost. Similarly, the optimization of power consumption is crucial for medical devices, such as electrically active implants [46], to enhance the longevity of the devices, and reduce the need for replacement surgeries.

Furthermore, optimization can be used for calibrating the model parameters to observed data [150]. A set of parameters is adjusted so that the agreement of the model with respect to

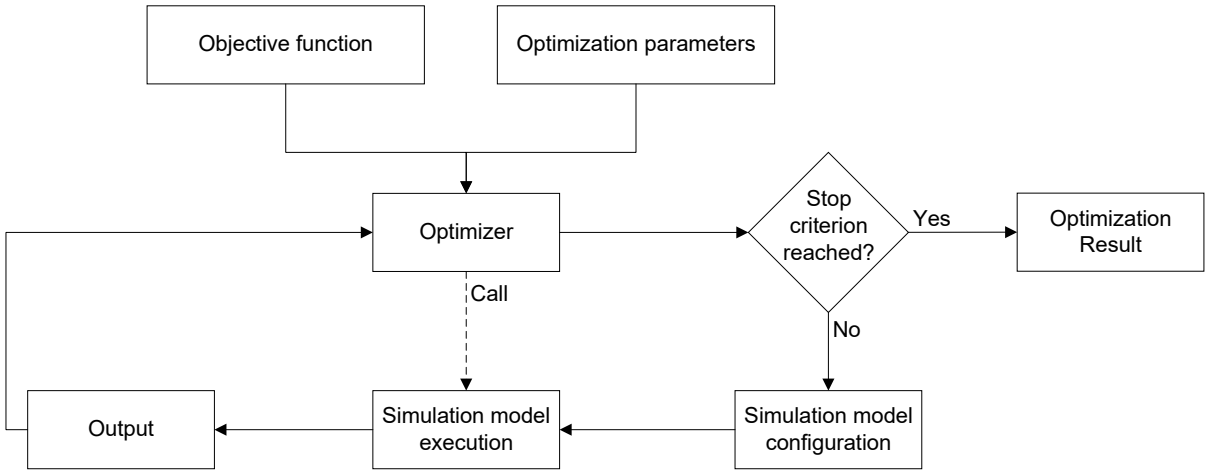


Figure 2.7: Simulation-based optimization loop, adapted from Nguyen et al. [149].

a set of experimental data is maximized, i.e., until the distance between simulation data and observed data falls below a defined threshold [151]. Various distance measures can be employed to define the discrepancy between simulated and observed data, as discussed above in the context of parameter estimation.

Optimization methods are categorized according to various reviews [152, 153, 148, 154, 155, 146]. The categories identified by Amaran et al. include the following [146]: a) *ranking and selection* involves systematically evaluating and comparing alternative solutions to identify the best-performing option with high confidence; b) *response surface methodology* learns the input-output function to approximate the simulation model by a surrogate. Subsequently, derivative-based optimization techniques can be applied; c) *gradient-based methods* estimate a gradient by means of finite differences, and then descend in the direction of the steepest gradient; d) *direct search* assesses candidate solutions generated by a certain strategy, using direct comparison of function values without approximating derivatives; e) *model-based methods* construct a probability distribution over the space of solutions to steer the search process.

It is important to note that there is no one-size-fits-all approach for all problems. And although some guidelines can be defined [146], what method and hyperparameters to use for a given problem class remains a question for further research [154]. In particular, how good a method performs depends on many properties, e.g., the number of factors involved and resulting discrete or continuous search space, or how expensive the simulation is [155]. Furthermore, properties of the response surface play a crucial role, such as whether it is a linear or complex nonlinear problem. But those are typically not known in advance [156].

Therefore, as discussed earlier, for many applications, using a method that finds good solutions within a reasonable amount of time may suffice [157]. Such robust solutions may be generated by taking into account methodology for surrogate modeling as well as experimental design [157], and accounting for uncertainties during the optimization [154]. For example, city planners may focus on reliable and resilient urban transportation systems that are capable of handling varying demand as well as accidents [158] instead of finding the one “best” solution.

Statistical Model Checking

Statistical model checking (SMC) is an approach employed to evaluate and validate specific properties of a simulation model, typically defined using temporal logic [159]. SMC involves a three-step process, as shown in Figure 2.8. First, execution traces of a stochastic model are sampled to capture the variability in its behavior and potential outcomes. Then, the given property specification is evaluated for each trace using a property checker (i.e., producing a

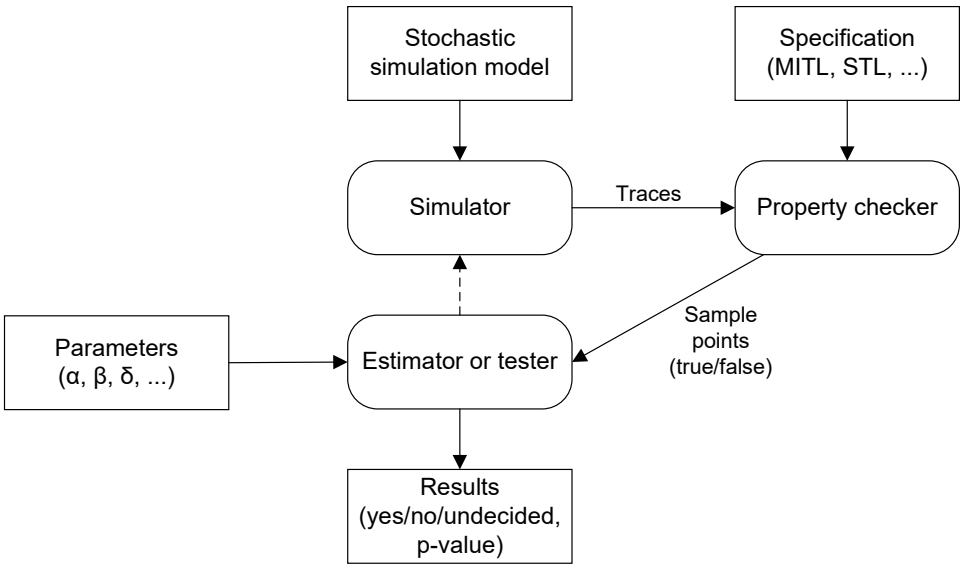


Figure 2.8: Workflow of a statistical model checking experiment, adapted from Agha and Palmskog [159].

sequence of true/false decisions). Subsequently, statistical inference is applied to analyze the collected sample points of the Bernoulli trial. Hypothesis testing is used to determine whether or how well the observations conform to a given requirement specification. The result of this comparison will either strengthen or weaken the confidence in the tested hypothesis.

Statistical model checking plays a crucial role in the validation of stochastic models. Requirements define desired properties of the model output using temporal logic. In the field of cell biology, e.g., the focus is often on reproducing specific trajectories that have been described in a scientific publication or observed in the wet lab. For instance, one might infer from data that there should be a peak in the concentration of β-catenin around simulation time $t = 600$ in the membrane model, and express this as temporal logic formula. By employing SMC, this behavioral requirements can be checked regularly, resembling a form of regression testing. This helps to ensure that all other analyses are conducted using a valid model, as the model's validity may change after refinement steps. Other requirements may refer to the runtime of the simulations stating, e.g., that each stochastic replication needs to complete within a specified time frame, such as less than 1 hour. SMC can also be employed to validate design requirements, e.g., ensuring that the battery runtime of a prototypical neurostimulator is at least 500 hours, or detecting faults in the designed circuits. Additionally, statistical model checking can address important research questions about the properties of the modeled system, which may be transient as well as steady-state properties [160]. In performance analysis, for example, SMC can answer questions about quality of service properties like expected latency and throughput in a network. Moreover, statistical model checking has relevance in analyzing safety properties of distributed real-time systems, including cyber-physical systems [161]. There it aids in the identification and assessment of potential safety risks, allowing for early detection and mitigation of issues that may arise. Recently, statistical model checking also has applications in model calibration, e.g., in combination with Bayesian parameter estimation [162] or in a "smoothed model checking" approach [163], where the model parameters are tweaked until the formula is satisfied.

Generally, there are two ways for conducting the hypothesis test. Either 1) a probability is given, and the model checker is asked to confirm that the model aligns with the requirement specification with the specified probability, denoted by p, or 2) the model checker is asked to determine the maximum probability that allows for accepting the hypothesis.

In addition, there are two approaches to sampling and evaluating the simulation traces. The first

approach uses statistical hypothesis testing, which involves generating all the required samples before applying a statistical test. Given a simulation model and a requirement specification, the property checker first computes the satisfaction of the desired property for all samples at once. Then a statistical test is performed, and the answer of the test is either true or false, with a p-value as a quantitative measure of confidence of satisfaction or violation, or undecided [164]. In the former two cases, the model checker's answer can be guaranteed to be correct within predetermined error bounds α and β for type I and type II errors, respectively [165]. In the latter case, p falls into the indifference region $[\theta - \delta, \theta + \delta]$ so that the test cannot provide error guarantees and the result therefore is undecided. One advantage of this procedure is that the sampling and hypothesis testing can easily be done in parallel. It requires, however, a fixed sample size n to be chosen beforehand.

The second approach to statistical model checking is incremental hypothesis testing. This test does not use a predetermined number of samples but instead determines after each batch of samples if additional samples are needed, or if the information currently available is sufficient to accept or refute the hypothesis. In Figure 2.8, additional iterations are indicated by the dashed backarrow. A popular method exploiting this sequential paradigm is the Sequential Probability Ratio Test (SPRT) [166]. It uses a Bayesian likelihood ratio that expresses the ratio of the likelihood of the data observed so far under the alternative hypothesis H_1 divided by the likelihood of the data under the null hypothesis H_0. After m observations have been made in the SPRT, i.e., after data points $y_1, \ldots, y_m$ have been obtained, the likelihood ratio is defined as

$$\frac{p_{1m}}{p_{0_m}} = \prod_{i=1}^{m} \frac{\Pr[Y_i = y_i | p = p_1]}{\Pr[Y_i = y_i | p = p_0]} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}}$$

with $d_m = \sum_{i=1}^{m} x_i$, and p_0 and p_1 being user-defined probability thresholds [165, 159]. The null hypothesis is accepted or rejected according to the error bounds given by α and β.

To specify hypotheses on the simulation traces and to enable statistical model checking, a variety of temporal logics have been explored, such as Metric Interval Temporal Logic (MITL) [167], Signal Temporal Logic (STL) [168], and Bounded Linear-time Temporal Logic (BLTL) [169]. Moreover, specialized logics for spatio-temporal properties [170] and corresponding specialized model checkers have been developed. In addition, there are powerful statistical model checkers, such as MultiVeSTa [171], that offer a variety of logics to specify the properties, and interfaces to different simulators for producing the simulation traces.

What-If Analysis

Golfarelli et al. define what-if analysis "as a data-intensive simulation whose goal is to inspect the behavior of a complex system [...] under some given hypotheses (called scenarios)" [172].

It is important to distinguish what-if analysis from sensitivity or perturbation analysis, as they serve different purposes. What-if analysis acts as an alternative to traditional forecasting methods, aiming to compare specific scenarios and support stakeholders in their management decisions in the various domains. For instance, what-if studies have been employed in management of the COVID-19 pandemic to assess policy options as well as combinations of policy options and their potential impact on the spread of the disease [173].

What-if analysis is crucial for exploratory M&S. This approach involves developing multiple model variations that go beyond simple parameter changes, encompassing modifications to rules and species within the model. Thus, instead of relying on a single "monolithic" model, exploratory modeling embraces the concept of "selective resolution", utilizing multiple models with different levels of detail [174]. Therefore, explorative M&S are of particular interest in the area of socio-ecological systems, e.g., to understand the implications of different policies around climate change and sustainable development [175]. Another example are social simulation studies where, e.g., various theories regarding the underlying psychological mechanics of human

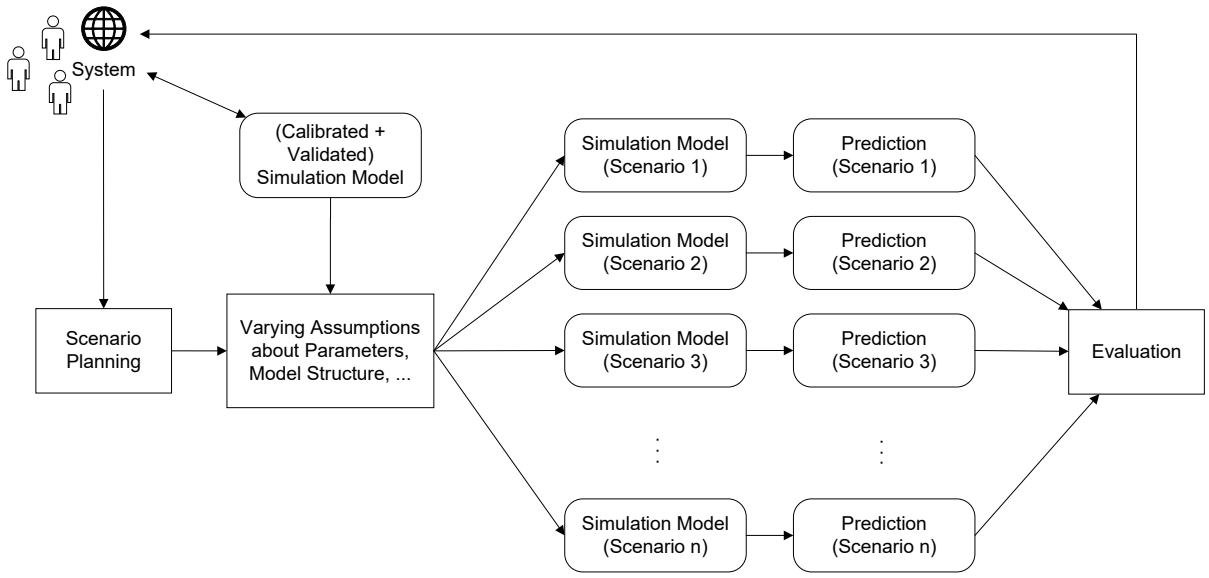


Figure 2.9: Process of planning, running, and evaluating what-if simulations.

migration can be implemented and compared, facilitating the understanding of these processes and their associated uncertainties [176].

Figure 2.9 illustrates the structure of a what-if analysis. During scenario planning, various aspects of the real world are thoroughly investigated to create multiple possible futures for exploration [177]. The created scenarios should either a) challenge assumptions about the future, b) provide an overall view of the environment and highlight interactions among trends and events, c) tie complex causalities together into a coherent and plausible manner, d) evaluate implications of possible future system discontinuities, e) look into the nature and timings of these implications, and f) shed light on the consequences of a particular choice or policy decision [177]. The scenario planning process, thus, creates varying assumptions about the parameters in a model as well as different model structures. Based on these selected alternative hypotheses, simulation models are constructed. Unlike an exhaustive screening approach, what-if analysis focuses on a subset of decision options, allowing for a more focused examination. Once the predictions for the alternative scenarios have been computed and evaluated, the best option can be selected, providing valuable feedback for guiding actions in the real system.

It is essential to have a calibrated and validated model as a starting point, which can then be adapted to the different scenarios [100]. What-if analyses, thus, usually take place in the later stages of a simulation study, after model building. Other simulation studies may already start with an existing model and their research question may be focused on making predictions using that model (see discussion of research questions in Section 2.1.1).

Predictions can focus on the evolution of variables over time within individual scenarios, capturing trends and patterns. Alternatively, specific time points in the future may be of interest. Additionally, steady-state behavior can be explored to understand the long-term behavior of the system. Furthermore, the analyses may need to consider different dimensions of the model output, to find a trade-off between several conflicting goals [178].

Beyond simple time course simulations that forecast the model behavior in time, what-if analysis may also explore model variants based on the alternative hypotheses using different types of simulation experiments [179]. Depending on the research question, one may be interested in the sensitivity or robustness of the alternative models, in explicitly checking properties using statistical model checking, or finding and comparing global minima via simulation-based optimization experiments.

Extensions of the procedure depicted in Figure 2.9 offer the capability to dynamically update

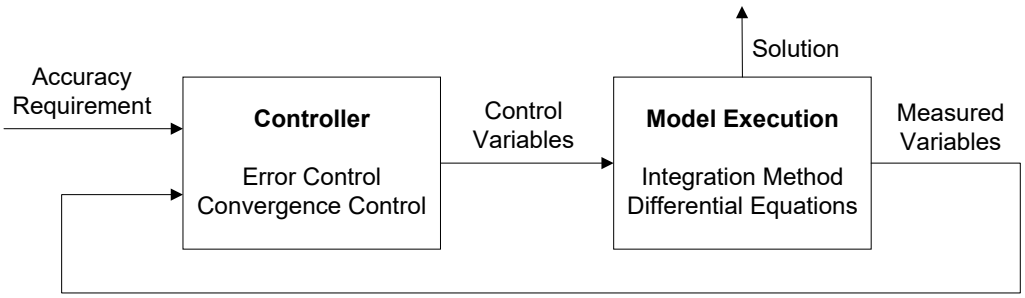


Figure 2.10: Error and convergence control feedback loop, adapted from Gustafsson [182].

predictions with new data. This interactive feedback is particularly evident in the case of digital twins, which represent a form of cyber-physical system. Digital twins continuously incorporate the current state of the modeled twin system into their predictions, allowing for real-time analyses and adjustments in the recommendations for action [180]. For example, by leveraging live energy monitoring data as well as a bidirectional communication with the physical world, up-to-date predictions can be made for various business strategies with respect to their energy efficiency [180]. Furthermore, the use of visual analytics can enhance the exploration and evaluation of alternative predictions. By employing interactive visual tools, analysts can actively examine the outcomes of various scenarios during the simulation execution, which enables them to discard certain scenarios early during the explorative process while at the same time zooming in closer on the more promising options [181].

Error and Convergence Control

Convergence testing is an experiment type that is crucial for retrieving meaningful results from numerical simulations. Controlling numerical methods is always a question of balancing accuracy and efficiency, and thus controlling the simulation error [182].

For simulations based on ordinary differential equations (ODEs) as well as the partial differential equations (PDEs) involved in FEA, a variety of solvers exist that incorporate an adaptive step-size regulation through feedback control. Still, depending on the characteristics of the problem at hand, some solvers might not present an adequate choice.

During the last decades, important theoretical works in the field of numerical analysis showed how different algorithms can be implemented for solving different kinds of ODE systems, e.g., stiff and non-stiff systems [183]. Despite the existence of such guidelines, from a practical point of view, convergence checks should be carried out in the early phase of the modeling process, before any actual experiments are conducted and data are produced with the chosen method. These checks typically involve face validating the trajectories, or setting tolerance thresholds for the simulation error.

Likewise, in the context of FEA, convergence tests are an important tool to investigate the numerical method's error in comparison to the true solution of the PDEs. The numerical error may stem from either the time discretization in the solver or from the spatial discretization of the finite element mesh [184]. Consequently, convergence studies may refer to checking that the selected solver method and its parameters do not affect the solution [185], or they may refer to checking the impact of mesh coarseness on the simulation results. Also round-off errors are listed as a potential source of errors; however, those are rarely significant on modern computer architectures [186].

Figure 2.10 depicts the general error and convergence control procedure. Given some error threshold, the controller adaptively adjusts the control variables, i.e., the size or number of the finite elements in the mesh or the step size of the numerical integrator. Using these variables, the simulation model is executed and some control measures are returned to the controller.

Computational results from previously obtained numerical solutions are then also taken into account for controlling the variables.

As part of this dissertation, only experiments for mesh convergence will be considered. Mesh convergence is known to be the largest source of error in numerical solutions of FEA, and it is difficult to estimate in most practical problems [186]. To derive an adequate mesh, an initial mesh is iteratively refined until the changes in the results fall below a specified threshold value. Global refinement applies to the entire mesh, for example, by subdividing every tetrahedron of a three-dimensional mesh uniformly into four congruent tetrahedra, and then checking how the target variable (e.g., the current in the electrodes) changes. In contrast, local refinement chooses the regions with the largest errors to refine them further in the next iteration.

Since mesh refinement strongly affects simulation runtime, automatic adaptive mesh refinement procedures aim to generate an optimal mesh with minimum number of elements [61]. This requires the definition of error estimates based on real-world measurements or analytical solutions [187, 188]. However, due to the lack of data, mesh refinement is often done manually via face validation. Also, there may be special cases where automatic refinement does not work, and parts of the mesh have to be treated manually.

Convergence testing (and adaptive mesh refinement) is a simple, computationally feasible approach to find solutions within specified error bounds, such as the L2 norm of the difference between successive solutions. In particular, simulation experiments for convergence testing allow checking whether the model and simulator setup are adequate to produce reliable results. Convergence testing thus is crucial for model verification, but can also be viewed as a kind of model calibration when configuring the coarseness of the mesh. For definitions of "verification", "calibration", and also "validation", see Section 2.1.2 and [3, 2].

2.3 Tools for Simulation Experiments

Although simulation models and experiments can be built using general purpose programming languages, most simulation studies today are implemented using a dedicated simulation software. Many of those tools provide multiple functionalities including particular domain-specific modeling means, and means for the design and conduction of simulation experiments. Advantages are reduced programming requirements, user guidance, graphical support and visualization. Tools like SESSL, on the other hand, are free referring to the modeling formalism, and instead focus on the specification and execution of simulation experiments. Similarly, SED-ML and others also focus on simulation experiments, however, they are not stand-alone tools but rather specification languages that need to be interpreted by additional software.

Consequently, there is a myriad of tools and languages out there, and their features regarding the types of simulation experiments supported vary. Table 2.1 lists a selection of different tools and specification languages. Note that the aim here is not to provide a complete overview but to demonstrate the variability of software solutions.

SESSL is an internal domain-specific language for experiment specification. It offers a variety of simulation experiment types. For some of them, SESSL provides own implementations, such as statistical model checking or local sensitivity analysis. Other experiment types are accessed through bindings to other tools, such as simulation-based optimization via Opt4J. James II, a plugin-based predessessor of SESSL, offers a subset of SESSL's features.

SAFE, developed in the context of the network simulator ns-3 [189], focuses on experiment designs for parameter scanning, including the entire pipeline from conception and design of simulation experiments to output analysis. NLRX is an R package for reproducible NetLogo model analyses. It also offers means for parameter scanning, e.g., Latin hypercube designs, and some methods for sensitivity analysis (Method of Morris and Sobol analysis).

SED-ML is an experiment specification language, developed in the context of computational biology, that not directly comes with means for experiment execution but can be imported by other

Table 2.1: Assortment of free and commercial modeling and simulation frameworks and experiment specification languages offering a variety of experiment types, including steady state estimation (SE), parameter scan (PS), sensitivity analysis (SA), parameter estimation (PE), optimization (O), statistical model checking (SMC), what-if analysis (WA), and error and convergence control (CC). Commercial software marked by *.

Framework/Language	Experiment Type							
	SE	PS	SA	PE	O	SMC	WA	CC
SESSL [36]	✓	✓	✓		✓	✓		
James II [190]	✓	✓	✓		✓			
SAFE [191]		✓						
NLRX [192]		✓	✓					
SED-ML [193]	✓	✓		✓				
COPASI [194]	✓	✓	✓	✓	✓			
C. E. P. Web Lab [195, 196]	✓	✓		✓			✓	
AnyLogic*[4]		✓	✓		✓		✓	
COSIDE*[5]		✓	✓		✓			
FEniCS[6]								✓

simulation software, such as COPASI, which then carry out the experiments. SED-ML allows specifying experiments for steady-state estimation, parameter scanning, and parameter estimation. Its XML-based syntax, however, is not intended for human specification but rather as an exchange format between simulation tools. Thus, the implementation of these experiment types depends on the specific tool interpreting the experiment specification. For instance, COPASI—a tool designed for modeling and simulation of biochemical reaction networks—is capable of executing these experiments. E.g., it estimates parameters by minimizing the weighted sum of squared differences between simulated and measured data [197]. Besides the least squares method, COPASI provides various other techniques for solving optimization problems (e.g., genetic algorithms and particle swarm algorithms). In addition, it offers means for local sensitivity analysis (including metabolic control analysis, a specific kind of sensitivity analysis involving perturbations of metabolite concentrations and reaction rates [198]).

The objective of the Cardiac Electrophysiology Web Lab is to exchange experiment specifications ("protocols") of various experiment types to test models in the domain of cardiac electrophysiology under a variety of experimental conditions. Protocols are encoded in an extension of SED-ML, and include steady-state analysis, time course analysis, and parameter estimation [199]. This aims to facilitate the comparison of results between similar simulation models answering common research questions in cardiac electrophysiology and to find errors in their implementations. The Web Lab thus may function as a benchmarking tool for alternative models built based on different hypotheses, thereby implicitly supporting what-if analyses.

Commercial tools (identified by an asterisk in the table) also support a broad range of simulation experiments. Anylogic, for instance, is a tool targeted at multi-method modeling including agent-based, discrete-event, and system dynamics modeling in various domains. It encompasses support for scanning single parameters, one-at-a-time sensitivity analysis and global optimization. Optimization may also be used for parameter estimation. Moreover, Anylogic allows specifying custom experiments via a dedicated Java class that can be enriched with own code. The custom experiments may, e.g., be used to create what-if scenarios via comparison of trajectories with different parameter settings.

[4] `https://www.anylogic.com/`, last accessed 19 July, 2024.
[5] `https://www.coseda-tech.com/coside-overview`, last accessed 19 July, 2024.
[6] `https://fenicsproject.org/`, last accessed 19 July, 2024.

Table 2.2: Tools and libraries for conducting specific experiment types, including steady state estimation (SE), parameter scans (PS), sensitivity analysis (SA), parameter estimation (PE), optimization (O), statistical model checking (SMC), what-if analysis (WA), and error and convergence control (CC).

Experiment Type	Tool or Library
SE	PySCeS[7], BioSwitch [200], ComplexLib [201]
PS	pyDOE[8], SciPy[9], lhs[10], pyLHD[11], pycubedoe[12]
SA	sensitivity[13], SALib[14], Gem-SA [202], uncertainPy [203]
PE	pyABC[15], ABCpy [204], PyBioNetFit [138], SBI [205], PEtab [206]
O	SciPy[16], optimx[17], Opt4J[18], SimOpt [207]
SMC	MultiVeStA [171], PLASMA-lab [208], PRISM [209], MRMC [210]
WA	–
CC	optimesh[19], pyGCS[20]

Other tools are rather tailored to supporting a specific modeling and simulation approach. The commercial simulation tool COSIDE, e.g., provides an environment for developing and analyzing models of heterogeneous hardware and software systems with SystemC and SystemC-AMS. Experiment types supported by COSIDE encompass parameter scanning, sensitivity analysis, and optimization. FEniCS, a free and open-source platform for the solution of partial differential equations, naturally supports mesh refinement algorithms for improving convergence of the models.

When looking at individual experiment types, there is an even more complex tooling landscape that needs to be navigated by users. A selection of tools and libraries are presented in Table 2.2. In case of parameter scanning, sensitivity analysis, parameter estimation, optimization, and statistical model checking, options are manifold. Methods for parameter estimation, for instance, are available in various Python packages. Additionally, there is the tabular specification and exchange format PEtab that can be used as input to several libraries, such as pyABC. It thereby facilitates reuse of parameter estimation experiments. In case of steady-state estimation and convergence control, fewer but specialized options exist. For what-if analysis, currently, no targeted tools are known.

Note that Tables 2.1 and 2.2 are simplified representations. They do not convey how these experiment types are implemented, or what specific methodologies are supported. E.g., pyLHD focuses on Latin hypercubes, whereas pyDOE includes also central-composite designs. For details about the various methods, the reader is referred to the overview given in Section 2.2.3 as well as the documentations and user manuals of said tools. Note also that these tables show a snapshot of the tools at the time of submission of this thesis, and features may evolve in the future.

[7] `https://pypi.org/project/pysces/`, last accessed 19 July, 2024.
[8] `https://pypi.org/project/pyDOE/`, last accessed 19 July, 2024.
[9] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.qmc.LatinHypercube.html`, last accessed 19 July, 2024.
[10] `https://cran.r-project.org/web/packages/lhs/index.html`, last accessed 19 July, 2024.
[11] `https://github.com/toledo60/pyLHD`, last accessed 19 July, 2024.
[12] `https://pypi.org/project/pycubedoe/`, last accessed 19 July, 2024.
[13] `https://cran.r-project.org/web/packages/sensitivity/index.html`, last accessed 19 July, 2024.
[14] `https://salib.readthedocs.io/en/latest/`, last accessed 19 July, 2024.
[15] `https://pyabc.readthedocs.io/en/latest/`, last accessed 19 July, 2024.
[16] `https://docs.scipy.org/doc/scipy/tutorial/optimize.html`, last accessed 19 July, 2024.
[17] `https://cran.r-project.org/web/packages/optimx/index.html`, last accessed 19 July, 2024.
[18] `https://sdarg.github.io/opt4j/`, last accessed 19 July, 2024.
[19] `https://github.com/meshpro/optimesh`, last accessed 19 July, 2024.
[20] `https://pypi.org/project/pyGCS/`, last accessed 19 July, 2024.

2.4 Computerized Support for Simulation Experiments

In theory, the goal of automating science has long been to "close the loop" between "examining the results" and "deciding what to try next" by harnessing advanced computational techniques [211]. Automating science thereby holds immense promise for accelerating scientific progress, driving innovation, and ultimately pushing the boundaries of human knowledge. In practice, however, the problem of automation needs to be divided into several subproblems and approached step by step, such as by developing techniques for streamlining repetitive tasks, analyzing vast amounts of data, generating hypotheses, and efficiently performing experiments. The same is true for the automation of simulation experiments.

In recent years, research on computerized support of simulation experiments has focused on several key areas. Among other challenges, Uhrmacher et al. identify the following research areas [212]:

Efficient execution and hardware utilization: One major concern of developments has been effectively utilizing distributed computing resources to accelerate the execution of large-scale simulation experiments [20, 21, 22]. Additionally, there has been a focus on adaptive parallelization of simulations in heterogeneous hardware environments [213]. By optimizing the utilization of available hardware resources, these approaches also facilitate the automation of complex simulation experiments.

Unambiguous specification: Another area of development has been establishing suitable means to unambiguously and machine-accessibly specify simulation experiments, which is a key prerequisite for providing computerized support. This has been achieved through scientific workflows [214, 215] and domain-specific languages and formats designed specifically for simulation experiments [24, 25, 192]. Additional languages have been proposed to specify various aspects of simulation experiments. For example, there are languages for specifying the simulation output, such as extracting summary statistics on model entities and executed transitions, as well as expressing observations based on specific conditions [216]. Moreover, specific types of experiments have been addressed, such as statistical model checking, which requires formally expressing behavioral properties in temporal and spatio-temporal logics [217, 218] or hypothesis languages [219].

Identifying core structures: Another key prerequisite for computerized support – and also for developing specification languages – is to identify the core structures of a simulation experiment. This includes the typical structures and concepts of specific experiment types (e.g., sensitivity analysis, simulation-based optimization, or statistical model checking), which have been represented via templates [220], schemas [221], or metamodels [26, 222]. To realize more flexible support for simulation experiments (e.g., including their translation between specification languages or the composition of experiment parts), Chapter 3 will explore the model-driven architecture and its value in automatically generating simulation experiments.

User guidance: Computerized support has also focused on guiding users through the entire process of conducting a simulation experiment, from setting up the experiment design to efficiently executing it, and finally analyzing and visualizing the results [191]. In [223], simulation studies were captured in a declarative artifact-based workflow, which allowed guiding users towards specific goals, e.g., when a validation or calibration experiment can be conducted or needs to be re-run when a milestone becomes invalidated.

Selection of methods and hyperparameters: Individual aspects of a simulation experiment have been supported computationally, e.g., by adaptively selecting a simulation algorithm during simulation execution using reinforcement learning [224], by composing algorithms to solve specific tasks through synthetic problem solvers [110], or by using algorithm portfolios [225]. The definition of hyperparameters has been addressed, particularly in the context of neural networks [226]. Similar approaches (e.g., based on advanced optimization algorithms) may be applied for parameterizing a simulation experiment for a given method and problem.

Automatic generation: First approaches for the automatic generation of simulation experiments have focused on generating hypothesis tests from formally specified hypotheses [227, 219]. In these approaches, no context information was used beyond these hypotheses and experiments were generated from scratch without considering previously executed simulation experiments. On the other hand, the strategy of reuse and adaption of simulation experiments requires some prior knowledge. For instance, the automatic composition of simulation experiments was realized based on formally specified simulation experiments and user-defined adaption steps [228, 229]. Others focused on the reuse of benchmark experiments for a specific type of model, and used a domain ontology to identify relevant benchmarks [195]. Additionally, simulation experiments could be automatically re-run within a workflow taking the user-specified requirements into account as well as the current milestones [52]. Table 5.1 in Chapter 5 provides further details about these approaches. However, a question remains as to how automatic experiment generation can be enhanced by incorporating context information about the entire story of the simulation study (i.e., the diverse activities conducted for model building, calibration, validation, analysis and the various artifacts involved), which will be subject of this dissertation.

2.5 Objective and Outline

Above, various areas of research were discussed that enable (e.g., unambiguous specification) or provide (e.g., user guidance) computerized support for simulation experiments. Among these approaches, the automatic generation of simulation experiments currently appears most promising for saving modelers time and effort in their simulation studies, and for allowing simulation studies to be conducted in a more systematic manner. Figure 2.11 illustrates how experiment generation can enhance the M&S life cycle.

Two strategies of experiment generation were identified: *from scratch* and *reuse and adapt*. An advantage of the reuse and adaption approach is that it simultaneously exploits and supports the iterative nature of simulation studies, where similar simulation experiments are frequently conducted, e.g., to re-calibrate or re-validate a simulation model.

Earlier approaches for the reuse and adaption of simulation experiments were based on precisely defined expectations about the type of experiment to be generated and the adaptions to be applied. However, how a simulation experiment needs to look like, when it needs to be generated, and also how to adapt it, really depends on the current context of the simulation study and should be derived from it. Context includes information about earlier model versions and experiments, the various artifacts of the conceptual model, such as requirements, assumptions, input data, and also the different activities in which these artifacts were produced or used, and the relations therein (provenance). So far, the full potential of the various context information available during a modeling and simulation study has not been explored. A challenge here is defining what belongs to "context" and how it can be represented in a machine-accessible manner.

Another challenge in automatically reusing and adapting simulation experiments is posed by the sheer diversity in simulation experiments as well as the heterogeneity in their specification. This chapter discussed the variability of simulation experiments across various dimensions: 1. the overall research question (e.g., understanding of the modeled system), 2. the roles of experiments in the modeling and simulation life cycle (e.g., calibration, validation and analysis), 3. the chosen M&S approach (e.g., discrete-event simulation), 4. the available experiment types and concrete analysis methods to be applied to the simulation model (e.g., statistical model checking with the Sequential Probability Ratio Test), and 5. the tools utilized for specifying and running the simulations and analyses (e.g., SESSL for both experiment specification and execution). Only if information is available about the criteria 1–5, suitable simulation experiments can be automatically generated.

Therefore, the objective of this dissertation is the development of a knowledge-driven process for automatically reusing and adapting simulation experiments that considers the variability

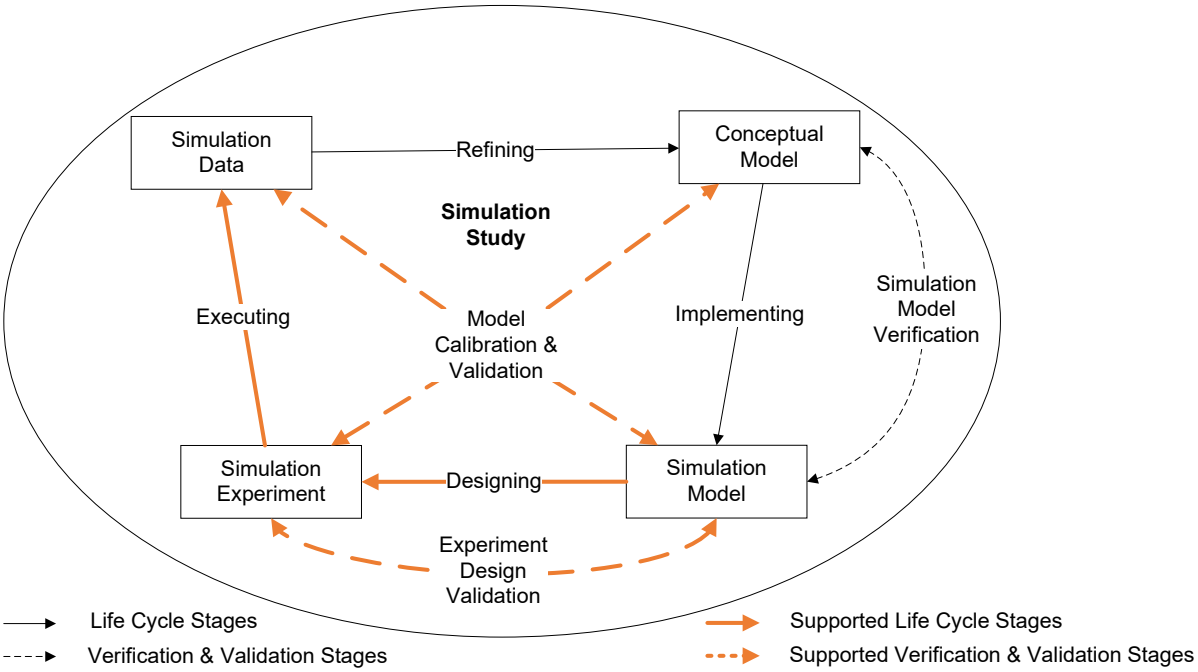


Figure 2.11: The modeling and simulation life cycle, showing the research focus of this dissertation. The highlighted arrows show the activities to be supported automatically. Adapted from Figure 2.1.

of simulation experiments and exploits context information about the simulation study. More precisely, the approach requires a) a deeper look into the expected ingredients of simulation experiments and means for explicitly specifying these and b) means for making the context of a simulation study (e.g., information about the modeled system, the research questions and assumptions, and the relation between the different modeling and analysis decisions) explicit, and c) a computational pipeline that integrates a) and b).

The outline of the dissertation is as follows. Chapter 3 concentrates on abstract and concise representation of experiment specifications via metamodels to facilitate their computational interpretability and reuse. Chapter 4 explores various sources of information from which experiment specifications can be derived. This includes the conceptual model, provenance of the simulation study, as well as ontologies. Finally, Chapter 5 integrates everything into a framework for the automatic reuse and generation of simulation experiments.

2.6 Summary

Simulation experiments play a crucial role in conducting successful simulation studies, particularly during the development of simulation models. They serve multiple purposes, including calibration, validation, and analysis of the simulation models. The specific roles of simulation experiments vary depending on the research question of the study. Moreover, throughout the life cycle of the study, different types of simulation experiments can be employed to achieve various objectives associated with the current stage of the simulation study.

The experiment types encompass estimating the steady state, scanning the parameter space, determining the sensitivity of model parameters, fitting parameter values to data, identifying optimal solutions, examining important behavioral properties, and making predictions for specific scenarios. The selection, specification, and execution of these experiment types depend on several factors that were discussed in this chapter. These factors include the current stage of the simulation study, the M&S approach being used, the concrete experiment methods available, and

the numerous software options for specifying and executing them.

Due to these many choices involved, conducting simulation experiments is a complex task that demands considerable time, effort, knowledge, and experience from the modelers. A simplistic "wind-it-up and let it run" approach will often not yield the best results in terms of driving the development of the model and gaining valuable insights. Consequently, the generation of simulation experiments presents an important methodology to support more efficient and error-free simulation studies.

This chapter provided an overview of computerized support for simulation experiments, including the automatic generation of simulation experiments. The primary research objective of this dissertation was identified: to develop an approach for the automatic generation of simulation experiments by reuse and adaption that integrates diverse context information about the simulation study to determine when to reuse specific experiments and how to adapt them to new situations.

Lastly, an outline was provided, giving a glimpse of the upcoming chapters in this dissertation and the solutions they will offer to enhance the field of conducting simulation experiments.

3 Making Simulation Experiments Explicit

3.1 Motivation

With new computational power, new availability of data, and the thereby strengthened role of simulation experiments in advancing science within the different application domains, the requirements referring to simulation studies and simulation experiments are becoming more rigorous, and the field of modeling and simulation is rapidly evolving. In particular, there are increasing requirements regarding the correct usage of sensitivity and uncertainty analysis [126], but also for calibration and validation to build realistic models [69], as well as increasing demand for open, replicable, and transparent methodologies [230]. In light of these new requirements, modelers that wish to create reproducible simulation experiments in an efficient and effective manner face new challenges. Especially, modelers working on interdisciplinary projects, where a variety of diverse simulation approaches and methods need to be brought together, would benefit from an easy-to-use methodology that supports them in specifying, executing, and managing simulation experiments. Also, engineers working on modeling and simulation software have to account for these new challenges and would benefit from a more structured and systematic approach for dealing with the increasing variety and complexity of simulation experiments.

The diversity and complexity of simulation experiments is, first of all, reflected in the various available specification languages (e.g., the Simulation Experiment Specification via a Scala Layer [36] and the Simulation Experiment Description Markup Language [193]), reporting guidelines (e.g., for finite element studies in biomechanics [62]), and frameworks (e.g., the Simulation Automation Framework for Experiments [191]). Although the simulation and experimentation tools to be used are typically preselected at the beginning of the study, having assistance in identifying an appropriate tool for specifying and conducting a specific simulation experiment can be valuable. Additionally, support for exploratory execution of all available experiment types within a chosen tool may be needed to gain an in detail view of a model's behavior. Moreover, support for comparing solutions across various tools is often necessary, particularly when conducting a cross-validation. E.g., to check that an extended Wnt simulation model [31] still exhibits the same behavior as the original model [32], the original experiments need to be executed in another tool (i.e., SESSL). Moreover, each type of simulation experiment may be realized with a variety of different analyses and methods, for instance, various full factorial or fractional factorial experiment designs for parameter scanning [120], global sensitivity analysis using the method of Morris [231] versus using Sobol indices [132] or partial rank correlation coefficients [232], or simulation-based optimization using various gradient-free methods such as hill climbing or particle-swarm optimization [156]. Thus, support for specifying and executing experiments using these different methods is desirable, which would provide users access to new methods, and methods they may not have been aware of before. And last but not least, there is the myriad of application domains, which require (partly) domain-specific solutions. In engineering disciplines, such as electromagnetics, for instance, the predominant simulation approach is finite element analysis (FEA), whereas in cell biology often discrete-event simulation (DES) is used, and consequently the simulation experiments (partly) look different in these domains. Thus, modelers as well as domain experts would benefit from assistance tailored to their application domain and simulation approach.

Model-driven-engineering (MDE)-based approaches can provide support in the specification, execution, and management of simulation experiments and resolve issues of interoperability,

reproducibility, and reusability beyond tools, simulation approaches, application domains, experiment types and methods. In an MDE approach, metamodels require certain inputs to a simulation experiment, and guide users through the experiment specification. Finally, combining the metamodels with means for code generation allows automatically generating executable simulation experiments [222].

Current MDE approaches for simulation experiments realize and demonstrate this for the generation of experiment designs for parameter scans [222] and hypothesis testing [227]. However, they support only these specific experiment types, concentrate on specific fields of simulation, or are tightly coupled with specific tools. As a consequence, their applicability and, thus, their impact on the way simulation experiments are conducted are limited. Furthermore, some typical benefits of MDE [233], such as improved knowledge sharing and further automation, are not yet available or not fully exploited for simulation experiments. Thus, a more general approach is desirable that can support simulation experiments more broadly, and thus, assist in conducting entire simulation studies more effectively and systematically.

In this chapter, a novel MDE framework is presented that enhances the state-of-the-art of conducting simulation experiments in the following ways:

- *Improved knowledge sharing across domains:* Experiment metamodels make knowledge about the structure and ingredients of simulation experiments explicit. By building various kinds of metamodels and storing them in a repository, modelers and developers are allowed to share knowledge about simulation experiments between the different modeling communities, e.g., FEA or DES. These usually develop their own simulation methodology and tooling independently, although the way simulation experiments are specified and conducted only partially differ and, thus, could be supported by the same experiment generation pipeline.

- *Increasing productivity and quality for complex experiments*: Especially for novice modelers, MDE clearly facilitates the specification of simulation experiments via graphical user interfaces (GUIs) and code generation tailored for a specific kind of simulation or type of experiment. Additional support is provided via an experiment validator, and support of complex simulation experiments is granted via metamodel composition. Due to the focus on complex experiments, also experienced modelers can benefit, and various types of simulation experiments and methods can be explored.

- *Reusability*: The metamodels give meaning to the ingredients of simulation experiments. Together with an easy-to-use, tool-independent specification format and a variety of backend bindings for translating and interpreting this specification, the flexible and automatic reuse of simulation experiments can be supported. This is of particular interest when the performance of different simulation tools needs to be compared [234], or model alternatives implemented using different modeling and simulation approaches shall be evaluated [235], or a cross-validation between two simulation models needs to be conducted [100].

- *Automation*: The presented MDE-based approach provides means for integration with other software, and thus presents the basis for further automation of simulation experiments. Automation of simulation experiments can be achieved if the knowledge about the various simulation experiments (given by metamodels) is put into relation with context knowledge about the simulation study. This may include knowledge about the early-stage products of the simulation study given by the conceptual model [82], knowledge about how products and activities are related given by provenance [43], or various other documentations [220, 236, 223], as discussed in Chapter 4. Here in Chapter 3, simulation experiments are automatically generated and executed based on the information collected in an artifact-based workflow. Later, in Chapter 5, simulation experiment specifications are automatically reused and adapted based on provenance information [237, 19].

The above features and benefits of the approach are demonstrated using five experiments from three actual simulation studies. The studies are part of the Collaborative Research Centre (CRC) 1270 ELAINE[21], which aims to develop novel electrically active implants for bone and cartilage regeneration as well as deep brain stimulation. As the research project is of interdisciplinary nature, in each of the three studies, a different modeling and simulation approach and application domain are in focus (i.e., stochastic discrete-event simulation (DES) of a cell signaling pathway, virtual prototyping of a neurostimulator, and FEA of electric fields), and various kinds of simulation experiments need to be conducted. For more background information on the different case studies, please refer to Section 1.2.

This chapter is a slightly modified version of the publication [17], where the model-driven approach for simulation experiments was presented. Earlier versions of the experiment generation pipeline and the vocabulary for specifying simulation experiments have been introduced in [221] and [220], and the concept of experiment schemas realized with JSON Schema in [221]. An ontology of sensitivity analysis methods was developed as part of [238]. The convergence study used in Section 3.5.4 to illustrate the integration of the MDE-based experiment generation with a workflow system was part of the publication [52].

3.2 State of the Art

3.2.1 Explicit Experiment Specification

The conduction of simulation experiments plays a central role in the modeling and simulation life cycle. Treating simulation experiments as first-class objects and making their specifications explicit facilitates their generation, reuse, repetition as well as their reproducibility, the latter having been identified as one of the main challenges in the computational sciences [239]. To support simulation experiments, a clear separation of concerns between simulation model, simulation experiment, and execution is necessary, which was already manifested in the concept of *experimental frames* [240]. An experimental frame constitutes the basic setup of a simulation experiment, including time base, input and variables. Based on this, e.g., an *experimental plan* can be created and executed [241]. The identification of further experimental frames, such as for parallel execution of experiments, processing and storage of output data, and graphical visualization of results, accelerated the development of various modeling and simulation frameworks, e.g., SAFE (Simulation Automation Framework for Experiments) [191] or the plugin-based JAMES II [190, 242]. In addition, following the reproducibility and credibility crisis of modeling and simulation [243, 11], a variety of approaches have been developed for capturing the various aspects of simulation experiments explicitly and machine-accessibly. The extension to *validity frames*, e.g., allows asserting the validity and reproducibility of simulation experiments by providing details about solvers and their configurations as well as the result collection [244].

Not surprisingly, also reporting guidelines for simulation studies increasingly demand information about simulation experiments. While many reporting guidelines clearly focus on the simulation model, e.g., the Overview, Design concepts and Details (ODD) protocol for agent-based models [75], the Minimum Model Reporting Requirements (MMRR), and the Preferred Model Reporting Requirements (PMRR) [245], they acknowledge that the model description should be followed by a documentation of the simulation experiments, e.g., in the "Materials and Methods" section of a research paper [75, sec. 5.6]. In contrast, STRESS (STrengthening the Reporting of Empirical Simulation Studies) aims to document the entire simulation study by emphasizing what a simulation study is about in terms of its objective, model logic, data, experimentation, implementation, and software availability [246]. TRACE (TRAnsparent and Comprehensive model Evaludation) also aims to document the entire simulation study but includes essential steps of the modeling and simulation life cycle such as calibration, analysis, and validation [236].

[21] `https://www.elaine.uni-rostock.de/en/`, last accessed 19 July, 2024.

Other reporting guidelines are domain-specific, e.g., for FEA studies in biomechanics [62] or social simulation [245].

However, the documentations based on reporting guidelines are typically provided as verbal narratives, and thus are not machine-readable, and the level of detail highly depends on the judgment of the user. The reporting guideline MIASE (Minimum Information About a Simulation Experiment), which directly targets the documentation of simulation experiments rather than entire simulation studies [247], circumvents this limitation by shipping with the external experiment specification language SED-ML (Simulation Experiment Description Markup Language) that implements the requirements with an own syntax and parser, and allows specifying simulation experiments unambiguously in a machine-accessible manner [193].

Besides SED-ML, which was developed in the context of systems biology, numerous domain-specific languages (DSLs) and formats for simulation experiments have been developed that all target different application domains or problem instances [248]. The Cardiac Electrophysiology Web Lab, e.g., focuses on comparing simulation models for a number of physiological conditions and therefore defines the experiments in an own experiment language based on the SED-ML format [195]. Other examples are the description language for the network simulator ns-3 (NEDL) [249], or Xperimenter for the design of experiments (DoE) [26]. These languages can be characterized as *external* DSLs, since they have a distinct syntax (and semantics) and require custom parsers and compilers for their execution.

In contrast to the mentioned DSLs, the *internal* DSL SESSL (Simulation Experiment Specification via a Scala Layer) is embedded in the general-purpose programming language Scala, and thus is a flexible approach that has the potential to support a variety of application domains and simulation tools as well as experiment types [36, 25]. NLRX, a specification format embedded in R, also supports various common experiment types [192].

Taking a different perspective, model-based approaches allow defining custom external DSLs using metamodels. From specifications in these "meta DSLs", executable code can be generated either in another external DSL or an internal DSL. Model-based approaches have been developed for the specification and generation of factorial experiment design [222] and various other experiment types [17], see Section 3.2.2. They also allow one to conquer a myriad of tools based on a single, tool-agnostic specification format if software bindings are implemented [36, 25, 17].

The development of approaches for explicit experiment specification is also the product of increasing awareness and application of the FAIR principles (Findable, Accessible, Interoperable, Reusable) [250]. These do not only apply to simulation experiments but to all artifacts of the scientific process. Also, these principles do not only refer to how simulation experiments are specified, but how they are made available. E.g., scripts for simulation experiments can be created and shared via Jupyter notebooks [251, 252], and the simulation models, experiments, and data can be bundled in reproducible containers such as COMBINE archives [253], Workflow Research Objects [254], or SciUnits [255]. With regards to the Findable and Accessible principles, these bundles can be shared via open model databases, e.g., BioModels [42] for models in systems biology, and CoMSES[22] (formerly known as OpenABM [256]) for agent-based models. The use of general web-based repositories with version (e.g., Github[23]), or services (e.g., ZENODO[24]) that assign a permanent address in the form of a digital object identifier (DOI) further contributes to the findability, accessibility, as well as reusability of simulation artifacts. Moreover, declarative, artifact-based workflows for entire simulation studies highlight the central role of simulation experiments by introducing them as research artifacts in their own right with their own life cycle and the stages *Choosing role* (e.g., calibration or validation), *Selecting approach* (for the specification and execution), *Choosing requirement* (to be checked during calibration or validation), *Specifying experiment*, and *Executing experiment* [223].

[22] CoMSES Computational Model Library `https://www.comses.net/`, last accessed 19 July, 2024.
[23] `https://github.com/`, last accessed 19 July, 2024.
[24] `https://zenodo.org/`, last accessed 19 July, 2024.

3.2.2 Model-driven Engineering and Generation of Simulation Experiments

Model-driven approaches add a layer of abstraction that allows describing simulation experiments in a tool- and language-agnostic manner. Metamodels capture the central concepts of a simulation experiment from which executable simulation experiments can be generated.

So far, model-driven approaches for modeling and simulation have mostly focused on the generation of (executable) simulation models [257, 258, 259, 260, 261], simulation on distributed architectures [262], or multi-formalism modeling [263].

With regards to simulation experiments, MDE has been applied in the context of experiment design and hypothesis testing. Teran-Somohano et al. [222] support the specification and execution of simple simulation experiments based on factorial designs, i.e., parameter scans. The approach by Dayıbaş et al. [26] is based on a DSL for experiment design and semi-automatic transformations to multiple execution platforms. Yilmaz et al. [227] propose the generation of experiment designs from hypotheses and outline an overarching goal–hypothesis–experiment framework.

Simulation experiments are also generated without explicit metamodels or MDE frameworks. These usually target specific problems within the modeling and simulation life cycle. Lorig [219], for instance, generates efficient experiment designs for hypothesis testing based on formally specified hypotheses. Peng et al. [228] support the successive composition of models by generating simulation experiments for composed models. More specifically, statistical model checking experiments of individual models are reused and adapted for the composed model based on explicit experiment specifications and explicit behavioral properties. Similarly, properties of extended or refined models can be checked [229]. Cooper et al. [195] focus on supporting the comparison of electrophysiology models using typical experiment types of that domain, i.e., time course analysis and steady-state analysis. Concrete experiment specifications (protocols) conducted with previous models are stored in an online database and can be repeated with new models. Ruscheinski et al. [220] present a template-based pipeline for generating a number of different experiment types including local sensitivity analysis, optimization, and statistical model checking. This pipeline was expanded by experiment schemas in Wilsdorf et al. [221] to support multiple simulation tools and approaches.

In this chapter, the latter experiment generation approach is further generalized using the MDE paradigm. The main novelties are a) the separation into two types of metamodels (base metamodels referring to the modeling approach and domain, and metamodels referring to the type of experiment) and a corresponding composition mechanism that allows specifying complex simulation experiments in a tool-agnostic manner, b) a metamodel repository, c) bi-directional transformations and execution bindings for a variety of M&S tools, and d) a command-line interface (CLI) for easy integration with other frameworks. As a result, the framework flexibly supports the conducting of diverse, complex experiment types, and it enables the sharing and reusing of knowledge even across different modeling domains, approaches and tools. Furthermore, the framework provides the foundation for automatically generating and reusing simulation experiments during the various phases of a simulation study, and thus for conducting entire simulation studies in a more systematic and effective manner.

The development of metamodels, as part of this MDE framework, is closely related to the development of ontologies as both represent knowledge in a structured manner. Ontologies for modeling and simulation have been developed to facilitate the data exchange, interoperability, and annotation of simulation resources [264]. Most ontologies define the components and concepts of simulation models instead of simulation experiments, e.g., the Discrete Event Simulation Component Ontology (DESC) [265] or the Discrete-event Modeling Ontology (DeMO) [266]. However, some ontologies overlap with metamodels for simulation experiments, e.g., the ontology for capturing physics-based models by Cheong and Butscher [267] and the here developed metamodel for FEA simulation studies (see Section 3.5) both include properties such as boundary condition and material properties.

3.3 Model-driven Approach for Conducting Simulation Experiments

In this chapter, a model-driven approach for conducting simulation experiments is developed. Its distinct features are the separation into two types of metamodels (domain metamodels and experiment type metamodels) and a corresponding composition mechanism, a metamodel repository, bi-directional transformations and execution bindings for a variety of tools, and a CLI for easy integration with other frameworks. The approach as a whole can improve knowledge sharing across domains and approaches, increase productivity and quality for complex experiments, make simulation experiments reusable, and automatically generate and execute simulation experiments in various settings.

3.3.1 Framework Overview

The approach for supporting the conducting of simulation experiments is based on the MDE principle. The central idea of MDE is combining metamodels (i.e., a structured implementation-agnostic representation of concepts) with a means for code generation [268]. Typically, MDE is realized in an architecture with four levels of abstraction and translations between them [269]. Figure 3.1 illustrates the four-level MDE approach for simulation experiments. Experiment *metamodels* of the various domains and approaches of modeling and simulation are represented by the level M_2. In addition to the definition of these *base experiments*, metamodels for a variety of *experiment types* (e.g., sensitivity analysis or simulation-based optimization) exist. A few will be presented in Section 3.5 and Appendix A. The two types of metamodels can be composed flexibly to create metamodels of complex simulation experiments.

How metamodels can be constructed is defined at the level of the *meta-metamodel* (M_3), for example, one could use formalisms such as EBNF [270], UML class diagrams [271] or schema languages [272, 273] to express the metamodels.

However, experiment metamodels are not only developed from scratch; therefore, they are stored in a repository from which parts can be reused by related areas of simulation or by other experiment types. The newly developed metamodels are again stored in the repository and thereby complement the knowledge collection about simulation experiments. A composed metamodel (consisting of a base experiment and the experiment type) can be loaded via an interface (GUI or CLI). The loaded metamodel dictates which inputs have to be provided to specify a valid simulation experiment. An experiment validator provides guidance for the modeler. If all inputs required by the metamodel have been collected, a concrete tool-independent experiment specification (*model* M_1) is generated. This general representation of the simulation experiment can now be easily reused for further experiment specifications, imported and executed by some tools directly, or used to generate a tool-specific experiment *instance* (M_0). To receive that tool-specific specification, the M_1-specification is automatically transformed into code of a target backend and subsequently executed using the respective tool binding. In the following, the distinct features of this MDE framework are described further.

3.3.2 Metamodeling Language

At the highest level of abstraction in the framework, the meta metamodel is situated. The meta metamodel is essentially the language in which the metamodels are specified. Meta metamodels are usually self-referential (i.e., they define themselves). As a result, no further layers above M_3 are required [269].

A metamodeling language needs to encompass a means for comfortably developing a new metamodel for a particular simulation domain or approach (base experiment) or a particular type of simulation experiment. The following requirements can be identified:

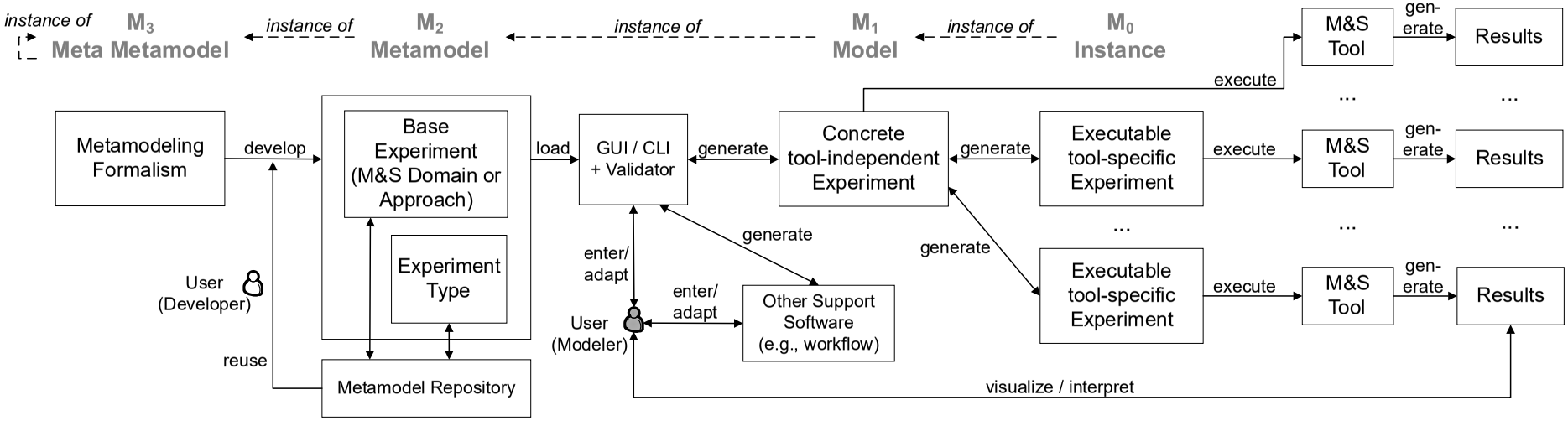


Figure 3.1: Four-layered MDE architecture for simulation experiments. The level of abstraction decreases from left to right. The experiment metamodels (consisting of the base experiment and experiment type; see layer M_2), developed based on a metamodeling formalism (layer M_3), are at the heart of the approach. Based on the metamodels, a tool-independent, exchangeable experiment specification can be generated (M_1), which then forms the basis to derive an executable, tool-specific specification (M_0). The generated experiment specification can then be automatically executed. However, the M_1 specification may also be imported and executed directly by some M&S tools. Additional support components (such as metamodel repositories, GUIs, and validators) assist users in conducting simulation experiments effectively, such that the remaining tasks concerning the user are entering or adapting information via the given interface and visualizing and interpreting the experiment results. The experiment generation pipeline may be used either as a stand-alone application or integrated with other support software via a CLI, e.g., a workflow.

- Hierarchical structuring via components and nested input properties;
- Unique names for components and input properties;
- Standard data types such as Boolean, string, integer, real, as well as arrays;
- Enumeration of admissible values and specification of default values;
- Maps to associate a key property with one or more other properties;
- Alternative specifications for input properties;
- Constraints to define simple type restrictions or the cardinality of an input property and, optionally, relations between multiple input properties.

UML [271] is widely used for metamodeling of software systems. UML class diagrams allow for a graphical notation that is intelligible to software developers as well as domain experts. The expressiveness of UML class diagrams is complemented with the Object Constraint Language[25] (OCL). Another way of metamodeling are schema languages such as XML Schema [272] or JSON Schema [273]. They describe the structure of text documents and, therefore, directly ship with a data exchange format (XML or JSON) for the layer M_1. They also encompass built-in means for defining simple constraints. For the detailed syntax and semantics of the constraint languages, the interested reader is referred to the OCL specification and the XML/JSON Schema specifications.

Both UML and JSON Schema are appropriate ways of specifying the experiment metamodels. With the workings of the model-driven approach in mind, it was decided to use JSON Schema in the following as it already comes with an established information exchange format (JSON) for the layer M_1. However, if conversions between the different metamodeling languages exist, the actual choice of formalism becomes less important [274].

3.3.3 Specification and Composition of Modeling Domains and Experiment Types

Metamodels describe the structure and ingredients of simulation experiment specifications. Two types of metamodels will be distinguished: *base* metamodels and *experiment type* metamodels. Base metamodels describe simple experiments (i.e., runs with a single parameter configuration) in a specific domain or modeling approach. Metamodels for experiment types, on the other hand, allow for supporting complex experiments where, e.g., parameters are varied, and specific analyses are conducted on the outputs.

A base metamodel for a fictive domain can be seen in Listing 3.1 (metamodels for experiment types can be created analogously). The metamodel is defined using JSON Schema and contains all the necessary information in one hierarchically structured file. It structures simulation experiments of the fictive domain into a model component, a simulation component, and an observation component. Each component encloses various input properties, characterized by type, information, choices, default values, and constraints. For example, the constraint `exclusiveMinimum: 0` was added inside the property `replications`. To express which inputs are required for specifying a valid simulation experiment, the keyword `required` can be used. Furthermore, default values can be expressed explicitly, e.g., for setting the stochastic simulation algorithm (`SSA`) as the standard simulator type of the domain. Moreover, to express alternatives, the keyword `oneOf` is used, as in the case of `stopCondition`, which can be given by either a specific stop time or an expression on the model output. To build a valid simulation experiment that conforms to the metamodel, exactly one of these alternatives needs to be applied. However, to keep the example short, these options are not specified further, and also the details of the observation component are omitted.

In fact, specifying the elements of the observation component is particularly tricky: what can be observed during a simulation and how to specify it depends on application domain or

[25] Object Constraint Language `https://www.omg.org/spec/OCL/`, last accessed 19 July.

is even intrinsic to the specific simulation model at hand (e.g., counts of the specific model entities and their attribute values, number of reactions executed, or summary statistics). What is possible to observe also depends on the simulation tool used and the interfaces it provides, e.g., specific expressions may be required, or observations may be flexibly coded in a general-purpose programming language [216]. As a result, defining a general language for simulation observation remains elusive. Attempts towards a flexible language for simulation observation draw on the concept of the database query language SQL [216]. The metamodels described below are designed to offer an interface for passing arbitrary expressions to the simulation tool, without performing additional checks on the provided expressions. In the future, the metamodels may be enhanced to support specific observation languages.

For comparison, another version of the base metamodel defined in a UML class diagram, thus using UML as the metamodeling language, can be seen in Figure 3.2. Components are represented by classes and the composition association, input properties by class attributes. The available choices for the simulator type are represented by an enumeration class. The alternative specifications for the stop condition of the simulation are represented by separate classes that inherit from the class `StopCondition`. For defining constraints such as default values or type restrictions, OCL is used. For example, the following invariant for the class `Simulation` can be specified: `inv: self.replications > 0`. This is, however, not visible at a glance in the visual representation of the metamodel. To sum up, both approaches are able to capture the information about this simple simulation domain in their own way. For the remainder of this dissertation JSON Schema will be used to define an external domain-specific language for specifying simulation experiments.

For specifying a simulation experiment, at least a base metamodel needs to be selected and filled with concrete input values by a user. However, the base metamodels can also be flexibly composed with the various metamodels for experiment types to create a metamodel for the current experimentation task at hand (such as sensitivity analysis, simulation-based optimization, statistical model checking, parameter estimation, etc., see Appendix A). Using the composition mechanism, complex simulation experiments can be supported for any modeling approach or domain. This is possible due to the orthogonality of the base experiment specification and the experiment type specification. Note that this dissertation focuses on the main experiment types and their methods (see Chapter 2), and does not discuss further analyses (i.e., post-processing or plotting of the results). However, this could be added as another component in the metamodels. The language SED-ML [193], for instance, allows the specification of various plot types including axis labels, etc.

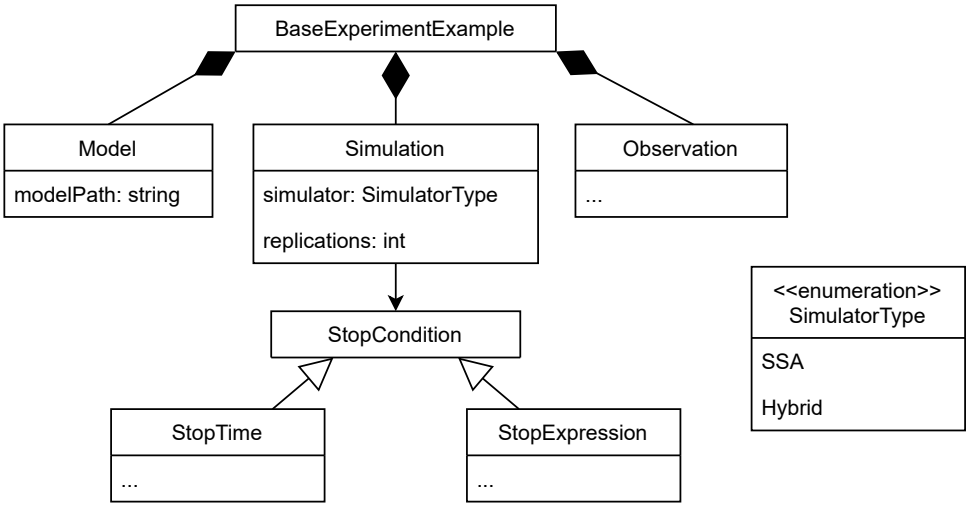


Figure 3.2: The base metamodel of Listing 3.1 defined in UML.

Listing 3.1: Base metamodel of a fictive modeling domain defined in JSON Schema.

```
{
  "$id" = "BaseExperimentExample",
  "properties": {
    "model": {
      "properties": {
        "modelPath": {"type": "string"}
      },
      "required": ["modelPath"]
    },
    "simulation": {
      "properties": {
        "simulator": {
          "type": "string",
          "enum": ["SSA", "Hybrid"],
          "default": "SSA"
        },
        "replications": {
          "type": "integer",
          "exclusiveMinimum": 0
        },
        "stopCondition": {
          "properties": {
            "oneOf": [
              "stopTime": {
                ...
      },
      "required": ["simulator", "replications", "stopCondition"]
    },
    "observation": {
      ...

```

3.3.4 Metamodel Repository

The metamodels are developed based on external knowledge from domain experts and potential users, and are stored in a metamodel repository. This gives users (modelers) access to a variety of base metamodels and experiment type metamodels that they can flexibly compose to execute complex simulation experiments, as discussed earlier.

New metamodels may be developed by modelers on-demand to fit their use cases. Throughout this process, modelers receive support from developers—persons with expertise in software engineering—who assist in the extension and composition of existing metamodels and crafting of entirely new ones. Looking ahead, the repository may be extended to provide additional software support for modelers in this process.

Often, depending on the needs of the new domain or approach or experiment type, not everything has to be developed from scratch, but metamodel components can be exchanged, added, or modified. Here, the reuse of knowledge about simulation experiments is not limited by specific domains and approaches. For example, research in the context of DES [240, 36] and computational biology [247] has identified common constituents of basic simulation experiment specifications: model configuration, simulation initialization, and observation. Those largely overlap with the metamodel for virtual prototyping, as described in Section 3.5.1. With respect to experiment types, metamodels can be extended to include more elaborate methods, e.g., new experiment designs or sensitivity indices. In the context of sensitivity analysis, the ontology developed in [238] may be a starting point. Additional information about the ontology and how to use it can be found in the excursus in Appendix B.

Also, properties may be reused across different experiment types. For instance, on the one hand, the specification of observed data and the specification of a distance measure between simulated and observed data can be found in simulation-based optimization experiments used to define an objective function for parameter fitting. On the other hand, the specification of

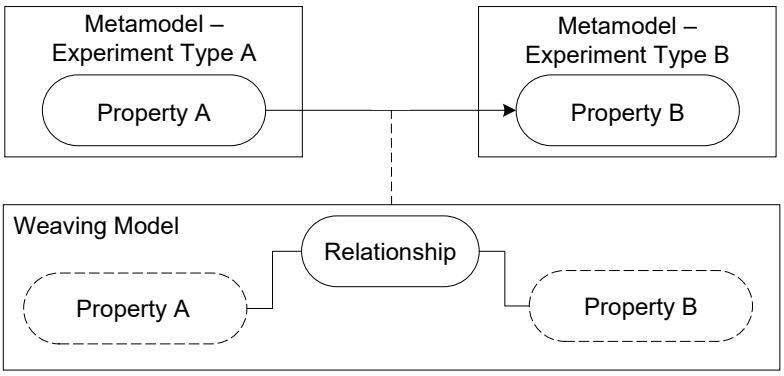


Figure 3.3: Dependency between two experiment metamodels defined by a weaving model. References are drawn with dashed lines. Adapted from Liu et al. [278].

observed data and distance measure may also serve as an alternative to a behavioral property provided as a temporal logic formula in a statistical model checking experiment.

In software engineering, weaving models allow specifying the reuse and composition of metamodel parts. They establish references between the elements of metamodels [275, 276], as illustrated in Figure 3.3. There, a dependency between two experiment types A and B, where a property A refers to a property B (e.g., a previously defined distance measure), is made explicit.

Together with means for version control and metamodel evolution, and information about the type of dependency, semantically meaningful reuse and composition of metamodel components can be achieved [277].

3.3.5 Interfaces

The MDE pipeline for simulation experiments can be used in two ways. The first option is a dynamic GUI: depending on the selected base metamodel and the experiment metamodel, a tailored GUI is generated to support modelers in specifying their simulation experiments. Figure 3.4 shows a screenshot of a GUI generated from the metamodel example shown in Listing 3.1 and a metamodel for sensitivity analysis. In the GUI, the metamodel components are displayed as individual tabs (“Model”, “Simulation”, “Observation”), input properties are displayed as rows, and alternative definitions are selected via drop-down menus, where, depending on the selection, other input properties become available as in the case of “Stop Condition”: either a stop time or an expression needs to be specified. The selection of the domain, simulation approach and experiment type takes place in the tab “Metamodel”. In this example, sensitivity analysis was selected as the experiment type, and thus an additional tab named “Sensitivity Analysis” was created (Figure 3.4). The tab “Backends” is used to select the target system for the code generation and execution.

As the second option for interacting with the experiment generator, a CLI is provided. It allows a flexible integration of the model-driven framework with other software. Via the CLI, the metamodels can be selected and composed, the experiment inputs can be passed and validated, the GUI can be opened, the target backend can be selected, and the experiment code can be generated or parsed. Consequently, one could integrate the framework with a procedure that automatically extracts experiment inputs from model documentation (e.g., parameter tables or conceptual model) [220, 19], or automatically generates simulation experiments from inside a workflow system (see Section 3.5). If not all input fields dictated by the metamodel can be filled automatically, the simulation experiment can be completed manually through the GUI.

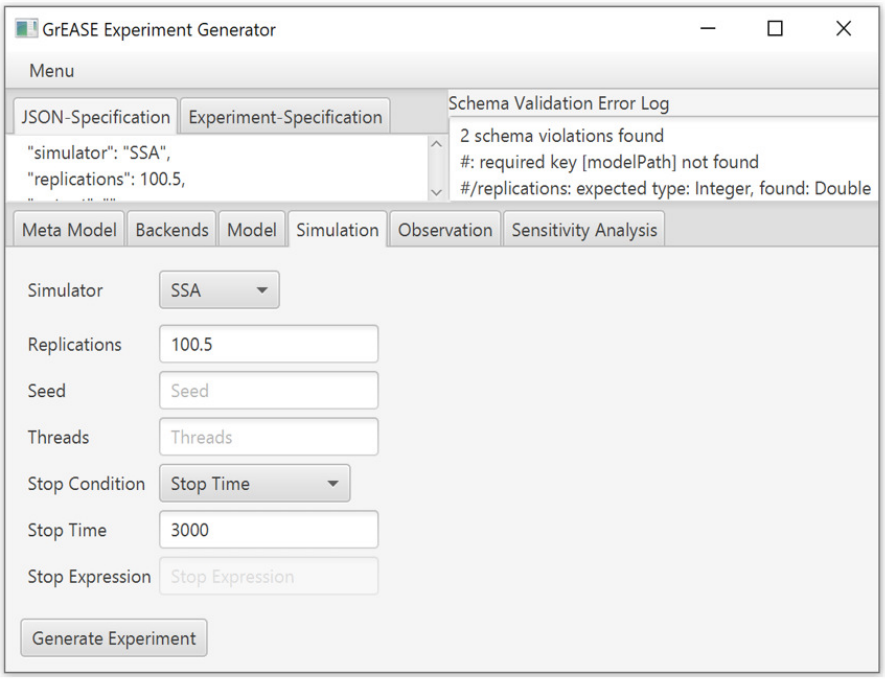


Figure 3.4: Screenshot of the experiment generation GUI. For a given metamodel, and depending on the provided inputs, new input fields are generated. Validation errors are reported at the top right if the inputs do not conform to the metamodel. In this example, a double value was entered for the number of replications but an integer value was expected. Furthermore, the model path has not been specified yet. The generated experiment specifications (tool-independent and tool-specific) are displayed at the top left. Executable, tool-specific specifications are only generated once all validation errors have been fixed.

3.3.6 Experiment Validation

Before the inputs collected via the GUI or CLI are passed on to the next layer (M_1) to produce a concrete tool-independent experiment specification, the entered values have to be validated. The validator checks conformance with the chosen metamodels, and thus, both structural checks and type checks are carried out. After each validation cycle, the validator gives immediate feedback to the user (see Figure 3.4) or the application, depending on which interface is used for the interaction. This step-by-step guidance supports inexperienced users primarily. But also experienced users can benefit, as they do not have to concern themselves with the intricate details of simulation experiment specifications and, instead, can concentrate on important tasks such as output analysis and result interpretation.

Listing 3.2 shows a created JSON document at the layer M_2. The document was validated according to the metamodel example described above. The validation was unsuccessful due to the missing property `modelPath` and due to using the wrong data type for the input property `replications`, see also Figure 3.4.

3.3.7 Bindings

The experiment metamodels provide a general vocabulary that the various modeling and simulation tools can refer to and communicate with. Metamodels are therefore one way for improving the interoperability of modeling and simulation software, and for guiding the choice of tools depending on what kind of simulation experiment was specified.

To go from an abstract experiment specification in a tool-independent format to executable

Listing 3.2: Simulation experiment filled according to the JSON metamodel defined in Listing 3.1. Validation errors in the specification are marked red.

```
1  {
2    "model": {
3      "modelPath":
4    },
5    "simulation": {
6      "simulator": "SSA",
7      "replications": 100.5,
8      "stopTime": 3000
9    },
10   ...
11 }
```

code, the metamodels have to be mapped to the syntax of the actual modeling and simulation tools. During this transformation step, also differences in the terminology need to be resolved, e.g., the concept of the stochastic simulation algorithm (SSA) has numerous implementations, which may furthermore be known under different names, such as "Next Reaction Method" (as used by COPASI [194]) versus "Gibson-Bruck Method" (as used by Cellware [279]).

Another important feature of the transformations is their bidirectionality, meaning that a concrete experiment specification in a specific language can also be transformed into a specification in the canonical format. This reverse generation requires capable parsers that have knowledge of the metamodels, and templates in the syntax of the given language [280]. The reverse generation may be applied, e.g., to execute the same or an adapted simulation experiment with a stochastic model either using SESSL [36] with the model defined in ML-Rules [35], using RNetLogo [281] with the model implemented in NetLogo [282], or SED-ML [193] with the model given in SBML [283]. For other simulation approaches also different choices exist, e.g., the free open-source software FEniCS[26] or the commercial software MATLAB[27] for finite element analysis.

Although it seems arduous to implement a transformation, it is more efficient than connecting all pairs of tools, with the experiment types and methods they provide, individually. Figure 3.5 sketches the binding of the presented MDE pipeline with external modeling and simulation software. Moreover, once agreed upon in the community, the structure and vocabulary of a metamodel are persistent and will rarely change, but rather be extended with new content, which will lead to some extensions in the transformations.

It might also be beneficial to maintain transformations to distinct versions of the same M&S tool or to legacy systems for keeping older simulation experiments reproducible and, for example, for testing the tool itself. In addition, the experiment metamodels provide guidelines for implementing new tools and, therefore, promote a more structured approach for developing modeling and simulation software.

In addition to the forward and reverse code transformations, the bindings allow automatically starting the generated experiments in the chosen backend. The backend and the implemented features within the backend then may allow the modeler to conduct further interventions during the experiment execution, such as pausing simulation runs or interactively zooming into parameter ranges (e.g., using visual analytics).

Note that a suitable backend could also be chosen automatically for a given task since the bindings make explicit which experiment types and methods are implemented where. Often, one wants to generate code for a single tool and, thus, in a single language. However, it is not uncommon to run the simulations in one tool, collect the results, and then, run the analysis (e.g., sensitivity analysis or parameter estimation) in another. The implemented transformation

[26] `https://fenicsproject.org/`, last accessed 19 July, 2024.
[27] `https://mathworks.com/products/matlab.html`, last accessed 19 July, 2024.

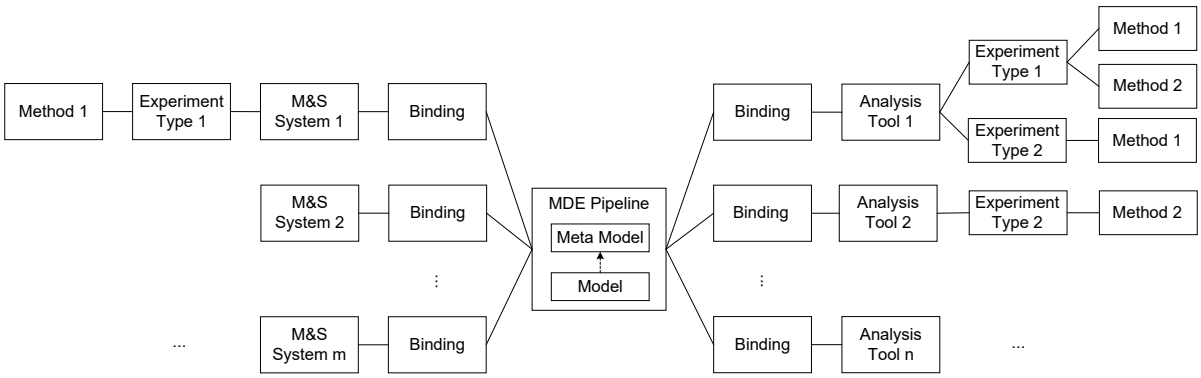


Figure 3.5: Bindings provide a mapping between a given metamodel (and thereby all models conforming to that metamodel) and the various modeling and simulation systems and analysis tools. The experiment types and methods offered by analysis tools can be flexibly combined with the simulation capabilities of the modeling and simulation systems. For instance, a parameter estimation experiment with the Python library pyABC may utilize the simulation software COPASI to execute the simulation model and produce output data. Additionally, some M&S systems offer own capabilities for conducting specific types of simulation experiments. For an overview of different tools for conducting simulation experiments, see Section 2.3. Figure adapted from Warnke and Uhrmacher [284].

mechanism supports the combination of suitable tools for simulation and analysis and, thus, allows generating a combination of scripts in different languages. Moreover, for some experiment types, such as sensitivity analysis, the toolchain can be divided further (see Section 3.5.2).

However, while the automatic execution of a simulation experiment clearly is tool-specific within the MDE approach, the experiment specification does not have to be. In particular, some tools may provide APIs for importing and executing simulation experiments. Thus, the general specification (e.g., given by the JSON models) could theoretically also be interpreted directly by the various tools. This is essentially how many tools in systems biology operate: they support the import and export of SBML models and SED-ML experiments. In this case, the layer M_0 of the MDE architecture would be skipped.

The generation of tool-specific experiment code, however, offers several advantages. For instance, domain-specific formats may be more readable for humans and therefore easier to customize during the semi-automatic generation procedure. Moreover, many existing modeling and simulation systems still expect input files in their specific formats.

3.4 Implementation

The model-driven framework for conducting simulation experiments is realized in Java 8. This proof-of-concept implementation, thus far, comprises metamodels for three simulation domains and different experiment types, the validation component, transformations and bindings to several backends, as well as the GUI and CLI. The source code is publicly available in a Git repository[28].

At the heart of the implementation are the simulation experiment metamodels, which are implemented using JSON Schema, Draft-07 [273]. JSON Schema was selected as a basis to formulate the metamodels, since over the past decade, after having been standardized in 2013, JSON[29] has become the most popular format for exchanging data on the web, and therefore, a variety of capable parsers and validators exist.

[28] `https://git.informatik.uni-rostock.de/mosi/exp-generation`, last accessed 19 July, 2024.
[29] ECMA-404 The JSON Data Interchange Standard, `https://www.json.org/`, last accessed 19 July, 2024.

When launching the application, the dynamic GUI, implemented using JavaFX, is opened by default. After the user has selected one of the included metamodels, the GUI generates the necessary input tabs and fields (as shown in Figure 3.4). When pressing the "Generate Experiment" button, all collected user inputs are stored in the multipurpose data interchange format JSON and presented to the user. The JSON files are validated against the JSON schemas using an open-source JSON Schema validator for Java, based on the org.json API[30]. Thereafter, the experiment specification for the selected backend is generated.

For the mapping between the general schema inputs in JSON and the syntax of actual modeling and simulation tools, the template engine FreeMarker[31] is used. FreeMarker has a built-in template language in which the mappings can be specified. For each backend and schema input property, a template fragment is implemented. A master template joins all the template fragments together. The template engine takes this master template and fills the template variables with the data from the JSON file.

As an alternative to the GUI, also a CLI is provided, which allows for even more flexibility when working with the MDE framework. For instance, one can transform an existing experiment specification for a target backend by running the command-line tool with the option `-t <originalFile> <targetBackend>`.

The backends and formats currently supported by the framework are SESSL/ML-Rules for simulation with R for experiment design methodology, SystemC-AMS with Uncertainpy, EMStimTools with Uncertainpy, and SED-ML. These will be applied in the evaluation (Section 3.5).

Regarding the runtime performance of the MDE pipeline, it is worthy to note that the time required for the experiment generation and transformation require less than a second on a standard laptop for the experiments shown in the evaluation. Thus, the generation time can be regarded as negligible in comparison to the time needed for experiment planning, input collection and result interpretation (which still have to be performed by the user and can only partly be automated) as well as experiment execution. In fact, the overall time to create a simulation experiment may be decreased by the presented approach as the user is not bothered with the syntactical intricacies of the experiment types in the various specification languages and tools.

3.5 Evaluation

Applying MDE for simulation experiments can improve and facilitate the building and sharing of knowledge, the productivity and quality of code, the reusability, as well as the automatic generation of simulation experiments. Thus, entire simulation studies can ultimately be conducted in a more effective and systematic manner.

The benefits of the MDE-based approach will be demonstrated using three real simulation studies from the interdisciplinary research project ELAINE on electrically active implants. These comprise the DES study of a cell signaling pathway by Haack et al. [34, 31], the virtual prototyping study of a neurostimulator by Heller et al. [46], and the FEA study of electric fields in a stimulation chamber by Zimmermann et al. [56]. The well-designed and realistic case studies are aimed at convincing modelers to adopt MDE for simulation experiments in their daily practice and to integrate this approach with other (existing) toolchains.

In the following, first, metamodels are developed to capture the characteristics of the three modeling and simulation approaches. Together with the domain modeling experts of the three studies (stochastic DES, virtual prototyping, and FEA) it is discussed how knowledge about simulation experiments can be exchanged within, but also between the different modeling and simulation communities. Next, it is demonstrated how, based on the base metamodels composed with a shared metamodel for the experiment type "global sensitivity analysis", three complex

[30] JSON schema validator. `https://github.com/everit-org/json-schema`, last accessed 19 July, 2024.

[31] Apache FreeMarker manual for Freemarker 2.3.30. `https://freemarker.apache.org/docs/versions_2_3_30.html`, last accessed 19 July, 2024.

simulation experiments can be specified in a straightforward manner to solve actual problems in the different application domains (analysis of a cell signaling pathway, the battery of a neurostimulator, and the electric field in a stimulation chamber). Then, it is shown how the JSON-based format can act as a standard exchange format to facilitate the automatic reuse of simulation experiments, e.g., for the cross-validation of two related cell signaling models. Finally, the approach is used to automatically generate simulation experiments by extracting information from a workflow to fill in a metamodel.

Note that all the described experiments are only snapshots of the three studies in the form of single simulation experiments. During the simulation studies, of course, further analysis steps with the same or other experiment types (e.g., simulation-based optimization or statistical model checking) are involved, until a validated model has been built and the initial research questions can be answered (see Chapter 2 about the modeling and simulation life cycle). For these further experimenting steps, the MDE approach can be applied analogously.

3.5.1 Improved Knowledge Sharing across Domains

The three simulation studies were conducted in the context of electrically active implants. However, they focused on different aspects involved in the development of such implants and, therefore, required different modeling and simulation approaches. Consequently, they required different metamodels for conducting simulation experiments. In this section, metamodels for the different approaches are introduced, i.e., the base experiments for stochastic DES, virtual prototyping of heterogeneous systems, and FEA in electromagnetics. This demonstrates the versatility of the approach and—most importantly—its value for improving knowledge sharing within and across the diverse domains and approaches of modeling and simulation.

Metamodel for Stochastic Discrete-Event Simulation

Stochastic DES is applied for modeling systems where the variables change at discrete time points, and the time of the next event is determined stochastically [38, 39]. In cell biology, e.g., modeling and simulating stochastic effects is of significant interest, especially for processes that involve low copy numbers [285]. Stochastic DESs are also increasingly applied in demography [286] or epidemiology [287] as an alternative to the traditional discrete time-stepped simulation approach with equidistant time increments.

Tables are used to represent the metamodels in a compact and readable way. The implementation as JSON Schema documents can be viewed in the source code repository. Table 3.1 shows the developed DES metamodel, which comprises three essential components, i.e., model configuration, simulation initialization, and observation (represented by sections in the table). In the metamodel, each component of a simulation experiment requires specific inputs (table rows). For instance, typical ingredients for a stochastic simulation are now made explicit, such as the number of replications, the random seed, and the number of parallel threads (see the simulation component). Each input is characterized by a unique name, a description, a data type, a set of choices, a default value, and information about whether this input is required or optional (table columns). E.g., the number of replications is a required property in stochastic simulation that has a default value of 1. Properties of type Map (e.g., configuration) assemble related inputs as key–value pairs. Keys are indicated by "κ" and values by "ν". Other properties (e.g., model, stopCondition, or observationTime) are of type Alternative and provide different ways of specifying a property. The alternative options are indicated by "|". For instance, the simulation model can be provided by either a folder and file name (local files) or a reference to an (online) resource.

The metamodel generalizes the structure and ingredients of the simulation experiments at the level of the modeling and simulation approach (i.e., DES). It can therefore be used for supporting

Table 3.1: Base metamodel for conducting experiments using stochastic discrete-event simulation. To create a valid simulation experiment, various information about the model, the simulation, and the observed quantities has to be provided. Each row describes an input property of the metamodel. Properties of type Map consist of a key ("κ") and values ("ν"). Alternative metamodel parts are indicated by "|".

	Name	Description	Type	Choices	Default	Required
Model	modelFile	Specify the simulation model	Alternative	–	–	yes
	| folder	Folder of the simulation model	String	–	–	yes
	fileName	Name of the simulation model	String	–	–	yes
	| reference	Reference to the simulation model	String	–	–	yes
	configuration	Configure the model parameters	Map	–	–	no
	κ parameterName	Input parameter name	String	–	–	no
	ν parameterValue	Input parameter value	Real	–	–	no
Simulation	simulator	Choose simulation algorithm	String	SSA, hybrid, ...	SSA	yes
	replications	Number of simulation replications	Integer, >0	–	1	yes
	randomSeed	Initialize random number generation	Long, >0	–	–	no
	parallelThreads	Number of parallel threads	Integer, >0	–	1	no
	stopCondition	Type of stop condition	Alternative	–	–	yes
	| stopTime	Stop at specific point of time	Real, >0	–	–	yes
	| stopExpression	Stop based on simulation state	String	–	–	yes
Observation	observables	Specify the observables	Map	–	–	yes
	κ observationExpression	Expression on model entities	String	–	–	yes
	ν observationAlias	Alias for observation expression	String	–	–	no
	observationTime	Choose option for observation	Alternative	–	–	yes
	| observationTimes	Observe at specific points of time	Array<Real>, >0	–	–	yes
	| observationRange	Observe time range and interval	Array<Real>, length=3	–	–	yes
	outputFormat	Choose reporting format	String	CSV, ...	–	no

the specification and execution of basic simulation experiments in various modeling domains, such as cell biology or digital circuits.

Metamodel for Virtual Prototyping of Heterogeneous Systems

Virtual prototyping allows for the design and development of products via modeling and simulation where building real prototypes is infeasible, e.g., due to ethical concerns or time restrictions, as in the case of neurostimulators for deep brain stimulation [46]. The fundamental paradigm for modeling and simulation of digital circuits is DES. This means that the knowledge about the ingredients of simulation experiments captured in the DES metamodel can be applied here as well. However, some virtual prototypes include components outside of the digital domain (e.g., to model the voltage levels of a battery) and, thus, require continuous-time representation. For these models, using the DES metamodel for the simulation experiments does not suffice. However, using the MDE framework, with relatively low effort, a new experiment metamodel can be defined for virtual prototyping of heterogeneous systems (with mixed digital and analog signals) based on the existing metamodel for experiments in DES. The model configuration component and the observation component can be reused from the shared metamodel repository as they are. Only the component regarding the simulation initialization needs to be adapted for the new simulation approach (see Table 3.2). The main difference is that, now, a fixed time step and time step unit are included, at which the discrete-time and continuous-time models are synchronized. Furthermore, the modified metamodel does not include the type of solver or scheduler explicitly, as this is usually assigned automatically to specific semantics defined in the language standard (e.g., SystemC-AMS [50]). Thus, the solver or scheduler are part of the simulation model and not changed in the experiments.

Metamodel for Finite Element Analysis in Electromagnetics

FEA is a general method that is capable of treating complex geometries and accurately computing, e.g., the properties and effects of electric fields in deep brain stimulation. A partial differential equation describing electric fields in the frequency domain is solved (physical model). Thus, time–harmonic fields are described and only the quasi-static behavior of the system under investigation is considered. The 3D geometry of the system is modeled explicitly, and corresponding boundary conditions and material properties are directly linked to the different domains of the geometric model.

As FEA is a completely different modeling and simulation approach, no parts from the previously defined base metamodels can be reused, and a new one is developed. Due to the separation of the geometric model and physical model, the experiment metamodel for FEA has two new components: geometric model and physical model (see Appendix A, Table A.1). The geometric model comprises the path to a mesh file or geometry file, and the number of dimensions. The physical model specifies the type of physics, material properties, boundary conditions, and the frequency for the quasi-static model. These are complemented by a simulation component that comprises information about the type of solver and the use of iterative mesh refinement to get a more accurate solution in certain areas of the geometry. For the observation component, observed properties can be specified, such as the electric field, and coordinates at which to evaluate these properties, as well as an output format.

Note that this first draft of the FEA experiment metamodel was developed with the background of electromagnetics in mind. Future efforts should aim to support FEA more generally. In particular, the various inputs and constraints depending on the type of physics need to be identified. In addition, multiphysics applications where mechanics, electromagnetics, and thermodynamics are coupled are of potential interest. In each discipline, PDEs comprising material properties together with geometrical constraints are the basis of the modeling approach [60]. Hence, the current metamodel can provide a basis for further discussions and adjustments.

Table 3.2: Modified metamodel component for virtual prototyping of heterogeneous systems. In contrast to simulations that are only based on discrete events, here also a time step has to be configured for the numerical integration. The rows describe the different input properties of the metamodel. Properties of type Map consist of a key ("κ") and values ("ν"). Alternative metamodel parts are indicated by "|".

	Name	**Description**	**Type**	**Choices**	**Default**	**Required**
Simulation	replications	Number of simulation replications	Integer, > 0	–	1	yes
	randomSeed	Initialize random number generation	Long, > 0	–	–	no
	parallelThreads	Number of parallel threads	Integer, > 0	–	1	no
	stopCondition	Type of stop condition	Alternative	–	–	yes
	| stopTime	Stop at specific point of time	Real, > 0	–	–	yes
	| stopExpression	Stop based on simulation state	String	–	–	yes
	timeStep	Time step of the simulator	Map	–	–	yes
	κ timeStepSize	Size of time step	Real	–	–	yes
	ν timeStepUnit	Unit of time step	String	s, ns, ...	–	yes

Table 3.3: Metamodel for the experiment type "global sensitivity analysis" (including information about default values and the required status). To create a valid sensitivity analysis experiment, various information about the model factors has to be provided, as well as information about the experiment design method and instructions for calculating the sensitivity indices. The rows describe the different input properties. Properties of type Map consist of a key ("κ") and values ("ν").

	Name	**Description**	**Type**	**Choices**	**Default**	**Required**
Experiment Design	factors	Information about the model factors	Map	–	–	yes
	κ factorName	Name of the factor	String	–	–	yes
	ν factorMinimumValue	Lower bound on the factor value	Real	–	–	yes
	ν factorMaximumValue	Upper bound on the factor value	Real	–	–	yes
	ν factorDistribution	Assumed distribution of the factor	String	Uniform, Normal, ...	Uniform	yes
	ν factorDistributionParameters	Parameterize the distribution	Map	–	–	no
	κ distributionParameterName	Parameter of the distribution	String	–	–	no
	ν distributionParameterValue	Initialize the distribution parameter	Real	–	–	no
	designMethod	Choose the experiment design method	String	MC, QMC, OLHC, PC, ...	MC	yes
	sampleSize	Number of samples	Integer	–	–	no
Index	indexType	Choose type of sensitivity index	String	Sobol, ...	–	yes
	bootstrapCI	Calculate confidence interval with bootstrapping	Boolean	–	false	no

3.5.2 Increasing Productivity and Quality for Complex Experiments

Besides sharing information about the base experiments, the MDE approach is designed to share metamodels for diverse, complex experiment types. Table 3.3, for instance, shows a metamodel for global sensitivity analysis (SA), which could be added to any of the above-described base metamodels. It comprises various important ingredients for the specification of a global SA, such as factor ranges and factor distributions, as well as a choice of different experiment design methods: Monte Carlo (MC), quasi-Monte Carlo (QMC) [288], orthogonal Latin hypercube ($OLHC$) [121], and polynomial chaos expansion (PC) [289]. For the first three strategies, the samples are used directly to calculate the sensitivity indices. With a PC, on the other hand, first, a surrogate model is constructed based on which the indices are computed. Different index types are offered. E.g., Sobol indices [290, 291] can be used, which are variance-based measures. The first-order Sobol index of factor X_i describes the individual contribution of this factor to the overall variance in the model output $V(Y)$:

$$S_i = \frac{V_{X_i}[E_{\boldsymbol{X}_{\sim i}}(Y|X_i)]}{V(Y)},$$

where X_i is the i-th input factor and $\boldsymbol{X}_{\sim i}$ denotes the matrix of all factors excluding X_i. The conditional expectation of the output Y is taken over all possible values of $\boldsymbol{X}_{\sim i}$ while keeping X_i fixed. The variance is taken over all possible values of X_i, and normalized by the total variance in the model output [291].

The total-order sensitivity index, T_i, accounts for all the contributions to the output variance due to factor X_i (i.e., first-order index plus effect of higher-order interactions with other factors):

$$T_i = \frac{E_{\boldsymbol{X}_{\sim i}}[V_{X_i}(Y|\boldsymbol{X}_{\sim i})]}{V(Y)},$$

or, stated differently, the total-order sensitivity index is the overall effect (which sums up to 1 for uncorrelated variables) minus the first order effects of $\boldsymbol{X}_{\sim i}$, i.e., the variance contribution of all terms in the variance decomposition excluding factor X_i [291]:

$$T_i = 1 - \frac{V_{\boldsymbol{X}_{\sim i}}[E_{X_i}(Y|\boldsymbol{X}_{\sim i})]}{V(Y)}.$$

For an overview of sensitivity analysis (SA) and the associated methods for sampling and calculating the indices, see Chapter 2.

Having made explicit the ingredients of global SA as a metamodel, simulation experiments to calculate Sobol indices for various simulation models can easily be specified. In the following, this is shown for three different models by composing the respective base metamodel with the SA metamodel. This has the potential to increase productivity during the simulation study since guidance is provided via a specialized GUI or CLI and input validation. Moreover, the intricacies of the code are abstracted away by the model-driven approach, and thus, the modeler does not have to worry about how to combine the simulation tools and analysis tools in a complex simulation experiment. Figure 3.6 shows three different ways of performing a Sobol analysis; they all can be generated using the same metamodel.

Sensitivity Analysis of a Wnt Signaling Model

To understand how electrically active implants affect the differentiation and proliferation of cells (and thus the composition and regeneration of tissue), the impact of external electric fields on central cellular signaling pathways (such as the Wnt/β-catenin signaling pathway) needs to be studied. Effects on membrane-related dynamics are of particular interest for the development of electrically active implants. Lipid rafts, specialized microdomains of the membrane, have

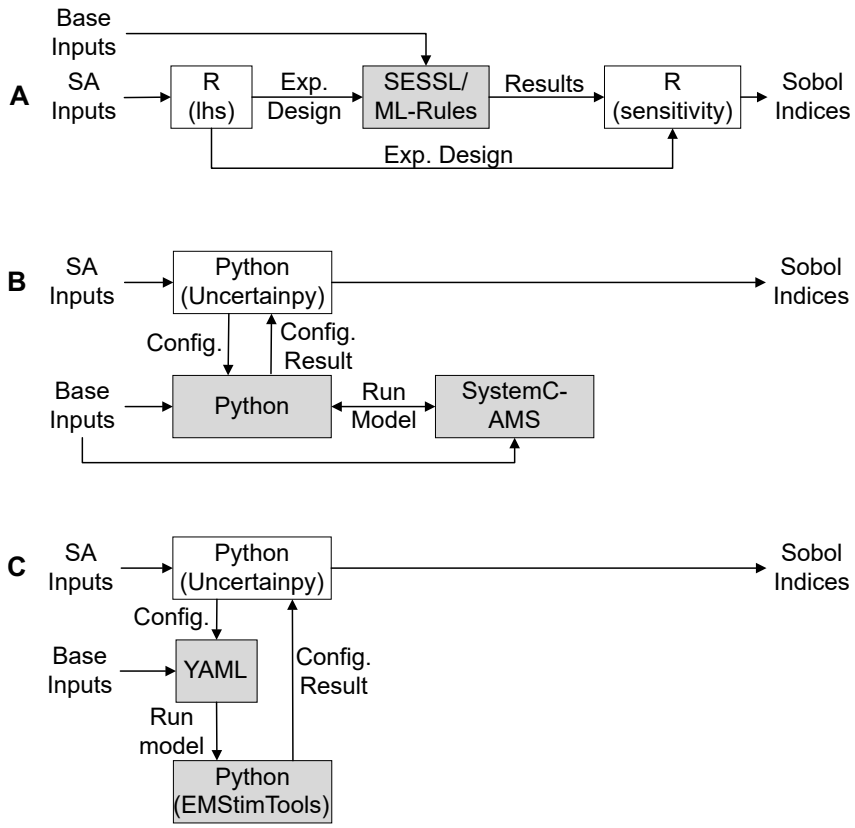


Figure 3.6: Schematic of the generated code for the three SA experiments for (A) the Wnt signaling model, (B) the battery model, and (C) the electric fields model. The experiments take the user-given base inputs and SA inputs and produce Sobol indices as outputs, using different combinations of M&S systems (grey boxes) and tools providing analysis methods (white boxes).

been found to sense the electric field and to direct the responses of cells [33]. Therefore, in the simulation study by Haack et al., a Wnt simulation model [31] was extended to capture both raft- and non-raft-associated endocytic processes of the Wnt/β-catenin receptor LRP6 in detail, including stochastic effects [34].

The model was written in ML-Rules [40] and simulated using stochastic DES[32]. The experiment metamodel for DES is available in the metamodel repository of the MDE framework and can therefore be used to guide the modeler through the specification of the base experiment, i.e., the constraints for one simulation run. Similarly, the metamodel for global SA is available to guide the modeler through the definition of the global SA. In the global SA, the modeler is interested in the sensitivity of the fraction of LRP6 receptors in the cell membrane, and therefore specifies this as the observed quantity of the experiment. The observation takes place after 60 minutes, i.e., at time point t=61. The sensitivity is analyzed with regard to three model parameters: $ke_{nonraft}$ and ke_{raft}, which represent the endocytosis rates of bound LRP6 receptor complexes inside and outside the lipid rafts, and the parameter $kLRAss$, which determines the rate at which the receptors shuttle into and out of the lipid rafts. For these three factors, the modeler specifies an experiment design with uniform distributions, as well as minimum and maximum values obtained from the literature [34], and enters the information via the generated GUI for the experiment type "sensitivity analysis". An OLHC design with 1750 samples is selected.

Once all inputs for the SA have been collected, executable experiment code can be generated,

[32]The Wnt model, the SA experiment, and the analysis results are available at `https://github.com/SFB-ELAINE/Case-Study-Endocytosis`, last accessed on 19 July, 2024.

i.e., a combination of an R script with a SESSL/ML-Rules script, as illustrated in Figure 3.6A. Without the experiment generation, the user would have had required expertise in writing both SESSL/ML-Rules scripts and R scripts, as well as expertise in the various types of SA and the respective libraries. Especially for novice modelers, correctly setting up a SA involving two specification languages can be challenging and error-prone, and therefore, quality and productivity can be improved by the MDE approach.

Since bindings are implemented to both SESSL/ML-Rules and R, the code of the composed experiment can not only be generated but also executed automatically. Figure 3.7A presents the results of the experiment to be interpreted by the user. They show almost no impact of the parameter ke_{raft} on the results. The variances of each of the other two parameters ($ke_{nonraft}$ and $kLRAss$) make up about half of the variance of the result.

The result indicates a slower pace of endocytosis within lipid rafts compared to LRP6 endocytosis independent of lipid rafts. The discrepancy can be attributed to the interaction between AXIN and LRP6, which is restricted to lipid rafts, and hence regulated by the parameter $kLRAss$. Bindings between AXIN and LRP6 are necessary for endocytosis within rafts, leading to a deceleration of raft-dependent endocytosis. In contrast, raft-independent endocytosis can occur without constraints. Consequently, endocytosis is mainly influenced by the parameters $ke_{nonraft}$ and $kLRAss$, whereas the impact of ke_{raft} is negligible.

Note that, for now, the automatic generation of plots from the analysis results is not supported; however, this could be included as a feature of this framework in the future.

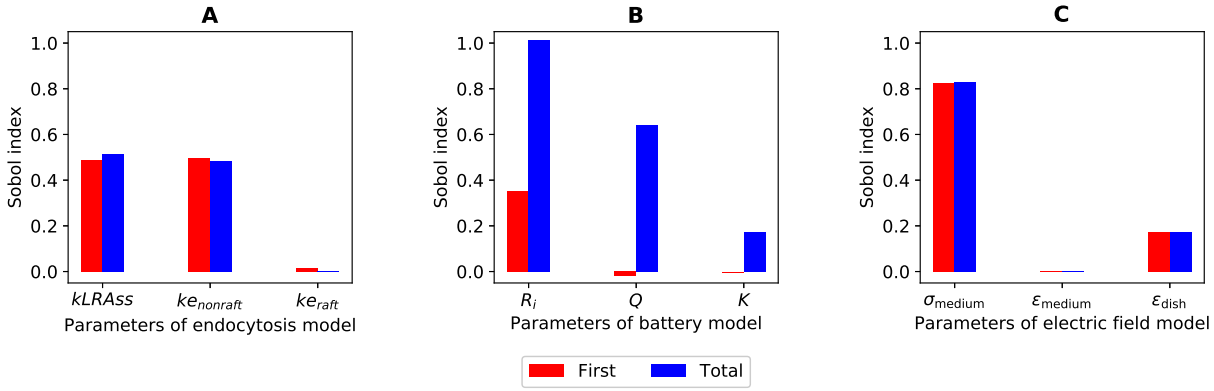


Figure 3.7: First- and total-order Sobol indices calculated for the three models. Whereas the Wnt model (**A**) and the electric fields model (**C**) show linear behaviors, the difference between the first- and total-order values of the battery model (**B**) indicates that the behavior of the model is governed by the interaction of parameters.

Sensitivity Analysis of a Battery Model

The underlying mechanisms of deep brain stimulation with electrically active implants are not fully understood. Since most stimulation experiments cannot be performed directly on humans, animal testings are necessary. However, neurostimulators designed for animal testing in rodents require a highly optimized level of miniaturization and power consumption compared to human implants. Therefore, virtual prototyping is used to design neurostimulators for rodents. The virtual prototyping study by Heller et al. [46] considers the interplay between various heterogeneous system components (i.e., battery, boost converter, microcontroller, and stimulation unit) and, thus, combines digital, as well as analog components. Whereas previous studies focused on the optimization of voltage levels inside the system, this study focuses on understanding the implications of different battery parameters on the overall runtime of the implant. The model is implemented in SystemC-AMS [50], an extension of the system description language SystemC,

and simulated using the hybrid simulator provided by SystemC-AMS, which accounts for both discrete and continuous signals[33].

The experiment metamodel for virtual prototyping is available in the metamodel repository. It was defined as a modification of the metamodel for stochastic discrete-event simulation (DES) to allow for a hybrid simulation with fixed time steps. The modeler uses the guidance of the model-driven framework to specify all required as well as some optional inputs for the base experiment, with the assistance of the validator. First, the model file is selected by providing the path to the SystemC-AMS project. Then, initial values are assigned for the parameters of the battery discharge equation [292]. In the simulation component, the number of stochastic replications is set to 100. The stopping conditions are specified as an expression on the model state: the simulation stops either when the battery reaches a specified runtime (`t >=` 2500 h) or when the main electrical parts fail due to low battery voltage (`V_boost ==` 0 V). The latter condition indicates that a complete functionality of the implant over the targeted runtime of 2500 h cannot be guaranteed. The time step and the observation interval of the simulation are defined as 1 s. The observed variables are the voltage and current levels to ensure implant functionality, and the overall battery runtime at the end of the simulation.

As they vary over different battery types, the internal resistance R_i, the battery capacity Q, and the polarization constant K are of special interest. To facilitate the analysis of these parameters, the modeler uses the model-driven framework and selects the metamodel for global SA (Table 3.3) to enhance the already specified base experiment. Parameter ranges and Gaussian distributions are requested for the three factors and entered by the user, and a QMC design is selected with 1000 samples. The QMC method is based on quasi-random low-discrepancy sequences, such as the Sobol sequence [293]. It is an efficient sampling method that possesses good uniformity properties by construction, which a LHC sample might not have or only after costly optimizations of the design, especially in high-dimensional spaces [294]. QMC therefore has often been the preferred method in engineering disciplines [295]. However, with new efficient methods for Latin hypercube sampling this advantage may dissipate [296]. A schematic of the code generated by the MDE approach is shown in Figure 3.6B. The backends used are the Python package Uncertainpy [203] and SystemC-AMS [50].

The generated experiment was started automatically via the backend binding; however, due to the complexity of the model, even after 12 h, the analysis was not completed. Therefore, the experiment was terminated by the user to find a more efficient approach. The metamodel for global SA provides a number of different experiment designs to choose from, and automatically requires all additional inputs and accounts for constraints associated with this other method. Thus, the MDE framework allows the user to quickly try out and compare different methods, e.g., also different sensitivity indices might be compared in the same way.

With the assistance of the model-driven framework, the experiment design could easily be substituted by a PC. PC builds an orthogonal polynomial representation of the model [289], i.e., a surrogate model. Coefficients scale the effect these polynomial terms have on the model output. To find the coefficients, mathematical techniques such as pseudo-spectral projection can be used. Once the coefficients are determined, evaluating the PC model is computationally inexpensive, and the sensitivity indices can be calculated without effort.

Using PC, the described experiment was automatically repeated and the runtime could be reduced to less than 3 h. The SA results, visualized by the user, are depicted in Figure 3.7B and show, due to small individual and large total effects, that the model response is strongly determined by interactions between the parameters. This is no surprise as battery discharge models are known to exhibit nonlinear behavior [292]. The internal resistance R_i seems to have a particularly strong effect on the overall battery runtime. Consequently, one should aim

[33] The described SA experiment and the analysis results are available at `https://github.com/SFB-ELAINE/Case-Study-Neurostimulator`, last accessed on 19 July, 2024. Unfortunately, the battery model itself cannot be provided, as it is part of a closed-source project.

for measuring this parameter in (real) experiments and pre-select batteries with lower internal resistance for further testing.

Sensitivity Analysis of an Electric Fields Model

The third simulation study aims to compute the electric field distribution in a specific chamber [56]. Based on the computed field distributions, the biological response of the stimulated biological sample can be linked to the electrical stimulation setup. A global SA is used to evaluate the influence of the dielectric parameters on the electric field strength at specific locations of the cells[34].

First, based on the metamodel for finite element analysis, the modeler configures a base experiment. The geometric model is loaded from a SALOME mesh file, and some parameters are configured such as the dimensionality. For the physical model, the type of physics is quasi-static, and Dirichlet boundary conditions are used. Material properties are specified for the materials *dish*, *medium*, *electrodes*, *air*, and *air gap*, and the frequency for the quasi-static model is specified. For the simulation component, a linear solver is selected. In the observation component, the electric field is specified as an observable, which is the negative gradient of the potential which is solved for.

Then, following the structure of the metamodel for global SA, the modeler enters all the required information about the factors and their value ranges into the GUI and assumes uniform distributions. The factors of the model refer to the various material properties, including the conductivity of the medium σ_{medium}, the permittivity of the medium ϵ_{medium}, and the permittivity of the dish ϵ_{dish}. A PC was chosen as the method for sensitivity analysis, as it is an often applied method in FEA, where simulation runs are computationally expensive.

Figure 3.6C shows a schematic of the generated code, which combines the Python packages EMStimTools [59] and Uncertainpy [203]. Again, the structured approach of this framework allows the user to quickly specify and execute the desired simulation experiment, even without prior knowledge of the Uncertainpy package, and methods typically applied in FEA (such as PC) can be easily accessed.

The first- and total-order Sobol indices are shown in Figure 3.7C. The results suggest that a change in the permittivity of the medium does not influence the field strength. Hence, the user concludes that the permittivity can be neglected in future uncertainty analyses for this kind of problem and similar input parameters.

3.5.3 Reusability

The final Wnt signaling model is the result of successively extending simpler model versions. The original Wnt model by Lee et al. (2003) [32] was extended by raft- and redox-dependent signaling events in a study by Haack et al. (2015) [31]. This new model was then extended further by endocytic processes in Haack et al. (2020) [34]. To ensure that the basic model behavior was not changed due to the extensions, various cross-validation experiments were required that compared the trajectories of the variables of interest.

When the study by Lee et al. was published, the model and experiments were custom coded and not shipped with the paper. Only years later, the original experiments were implemented in SED-ML [193] and the model in SBML [283], and shared via the BioModels database[35]. This now enables the reuse of these simulation experiments to be repeated with the extended model by Haack et al.

The Wnt model by Haack et al. (2015/2020), however, was specified using the rule-based modeling language ML-Rules [40], and the experiments were conducted using the experiment

[34]The model, the described SA experiment, and the analysis results are available at `https://github.com/j-zimmermann/EMStimTools/tree/master/examples/experimentSchemas`, last accessed 19 July, 2024.

[35]See BioModels [42] entry at `https://www.ebi.ac.uk/biomodels/BIOMD0000000658`, last accessed 19 July, 2024.

specification language SESSL [36]. Thus, in order to reproduce the results from the study by Lee et al., the SED-ML experiment specification needs to be adapted for the new model, and translated to SESSL. This can be supported by the model-driven approach.

Listing 3.3: Cross-validation experiment in the exchangeable JSON format, with the inputs (blue) already adapted for the Haack et al. model.

```
{
  "model": {
    "modelFile": {
      "folder": "models",
      "fileName": "M2_2.mlrj"
    }
  },
  "simulation": {
    "simulator": "SSA",
    "replications": 10,
    "stopCondition": {
      "stopTime": 960
    }
  },
  "observation": {
    "observables": {
      "observationExpression": ["Cell/Nuc/Bcat"]
    },
    "observationTime": {
      "observationRange": {
        "observationRangeStart": 0,
        "observationRangeEnd": 960,
        "observationRangeInterval": 6
      }
    }
  }
}
```

First, the metamodel for DES directs the automatic parsing of the original specification and provides meaning to the parts of the experiment specification. As the experiment type is a time course analysis, a base metamodel suffices to capture all the inputs. Once transformed to the quasi-standardized JSON format, the experiment specification can be adapted to work with the model by Haack et al., e.g., the file name and the simulation stop time (due to the use of different time scales) need to be changed. This can partly be performed automatically by exploiting additional knowledge in the form of ontologies. E.g., UniProt [297] provides a unique identifier for each protein and allows transforming the variable names in the observation expressions from one model to another.

Especially, these ontology-based automatic transformations are valuable for the reuse of simulation experiments as they prevent inconsistencies that can easily happen when translating an experiment for a different model or a different specification language. As mentioned, the variable names might have a different meaning in Model A compared to Model B, but also, e.g., the units used for the rate constants might differ. Incorrect translations may produce distorted simulation results and, therefore, seemingly different model behavior, leading to the wrong conclusion that the new model cannot be successfully cross-validated. A further discussion of necessary (automatic) adaptions can be found in Section 5.4.4.

The automatically parsed and transformed experiment specification is then presented to the user to ask for further adaptions or additional information that could not be extracted from the old specification. Listing 3.3 shows the finished experiment specification in the JSON-based exchange format after being adapted for the model by Haack et al. From this, the SESSL-specific experiment can be generated and executed automatically.

The result of the cross-validation is depicted in Figure 3.8. It compares the trajectories of the key protein β-catenin, an indicator of the pathway's activity, produced by the Lee et al. and the Haack et al. model (with an adapted time scale) when stimulated with a transient Wnt stimulus. Both β-catenin curves show the same peak at the same time. From this, the user concludes that

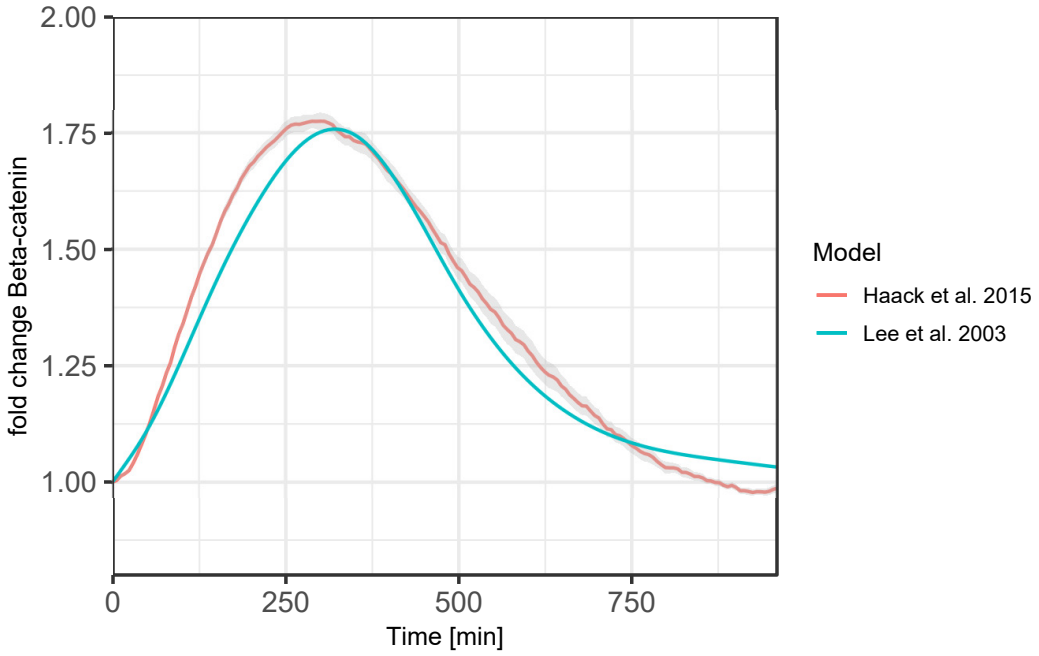


Figure 3.8: Results of the cross-validation of the Lee et al. model [32] and the Haack et al. model [31] (after applying a scaling factor of 0.28 to the β-catenin trajectory).

the extensions applied in the study by Haack et al. do not alter the central dynamics of the pathway, i.e., the model is valid.

3.5.4 Automation

In the previous subsection, one case of automation was already demonstrated. There, the MDE pipeline was the foundation for reusing simulation experiments automatically [237], and for automatically interpreting and translating experiment specifications via ontologies. Now, to go one step further, this approach can be integrated with other frameworks to support simulation studies as a whole, e.g., with an artifact-based workflow (an IDE that provides guidance throughout the modeling and simulation life cycle).

In an artifact-based workflow, the central products of a simulation study are identified and made explicit as artifacts. These include the conceptual model, the simulation models, and the simulation experiments. Each artifact is characterized by the stages a modeler can move through to achieve certain milestones and preconditions of the subsequent stages called guards [223]. Figure 3.9 shows the conceptual model artifact of an artifact-based workflow for FEA studies, as introduced by Ruscheinski et al. [52]. While moving through the stages of the conceptual model, the modeler specifies various metadata about the model, such as the modeling objective, requirements, and input data, which are stored inside the artifact. This metadata of the conceptual model artifact, as well as other artifacts, can be used in the automatic generation of simulation experiments. For instance, in [52], a FEA simulation study of an electrical stimulation chamber was conducted with assistance from the workflow. Once the user reaches the stage *Specifying simulation experiment*, the MDE-based experiment generation can be involved via its CLI. For instance, the information collected by the workflow can be passed to the MDE pipeline to automatically generate a convergence test for the model.

The convergence of numerical methods is crucial for FEA studies in order to retrieve meaningful results from a numerical simulation [61]. In a convergence experiment, the mesh is incrementally refined until the estimated discretization error lies below a given error threshold or until the maximum number of iterations is reached. E.g., a meshing algorithm, such as the Netgen

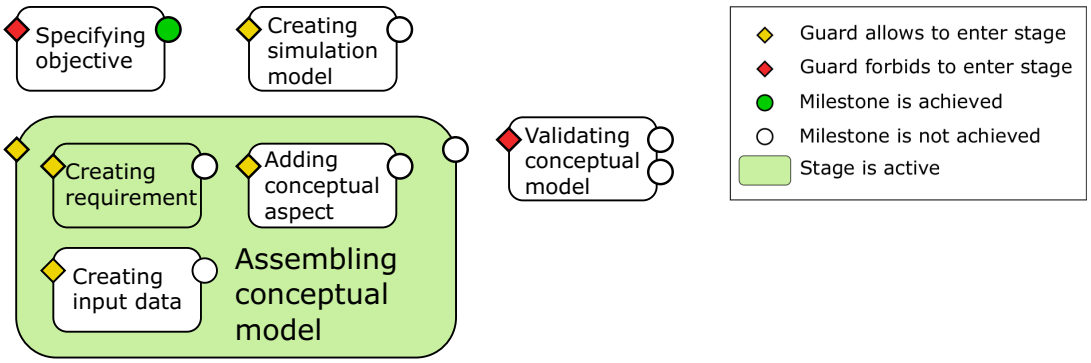


Figure 3.9: The conceptual model artifact of the artifact-based workflow with all its stages, guards, and milestones, adapted from [52]. Since the milestone of the *Specifying objective* stage has been achieved, two other stages can be entered: the modeler can start with creating the simulation model, or the conceptual model can be (further) assembled. In this example, the modeler enters the *Assembling conceptual model* stage and adds a new behavioral requirement. As long as the conceptual model is not fully assembled, it cannot be validated, and therefore, the guard of that stage is disabled.

algorithm [298], can be iteratively applied. A metamodel for convergence experiments, therefore, requires the following inputs:

- The region of interest;
- An error metric allowing to estimate the discretization error for a given region;
- The maximum number of iterations or an error threshold to control the error;
- An initial meshing hypothesis, i.e., the minimum and maximum size of the finite elements, to initialize the meshing algorithm.

For a detailed version of this metamodel, please see Table A.9 of Appendix A.

By connecting the command-line tool of the MDE framework to the workflow system, some of the inputs required by the metamodel can be filled automatically by extracting metadata from the artifacts, e.g., the region of interest and the error metric were specified as requirements in the conceptual model [223]. Other inputs are specific to convergence studies, e.g., the initial minimum and maximum size of the elements, and thus, either need to be entered manually by the user (e.g., by opening the GUI pre-filled with the extracted values), or are initialized with default values (specified in the experiment metamodel). Figure 3.10 shows the results of the convergence experiment that was generated semi-automatically for the model of the electrical stimulation chamber. It shows how the observed variable (current at Electrode 1) converges with increasing degrees of freedom (DOFs) in the finite element model. Here, the DOFs refer to the number of nodes in the mesh. After the fourth iteration, the refinement can terminate.

Note that the concept of automatically generating experiments and recycling information from other sources, such as workflow artifacts, is not restricted to the rather simple mesh refinement method used above. Other more sophisticated methods could be applied if the metamodel is extended accordingly. For example, one could specify areas of the geometry where a finer mesh is required, or regions where the electric field amplitude ranges within an interval that is considered therapeutically beneficial in deep brain stimulation [52].

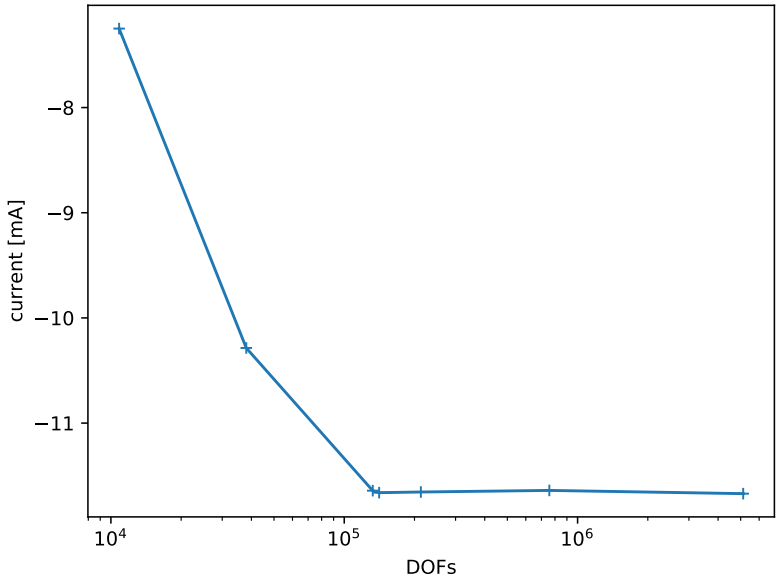


Figure 3.10: Convergence of the current with respect to the number of degrees of freedom (DOFs).

3.6 Summary

In this chapter, a model-driven framework for supporting the conducting of simulation experiments was presented. The approach is based on the development of metamodels, which structure knowledge about simulation experiments. Base metamodels capture knowledge about the various simulation domains and approaches, such as stochastic DES of cell signaling pathways, virtual prototyping of neurostimulators, and FEA of electric fields. Additional metamodels can structure knowledge about different types of simulation experiments. These can be composed with the base metamodels on demand, as demonstrated for a global sensitivity analysis. Storing the metamodels in a repository further facilitates knowledge structuring, as well as its usage and sharing across these various simulation domains and approaches.

Based on the metamodels, the MDE framework supports modelers in specifying simple and complex simulation experiments. It provides means for metamodel composition, guidance for specifying the concrete inputs for the different ingredients, and automatic validation of the inputs by checking approach- or experiment type-specific constraints and proposing available options, such as methods for sensitivity analysis. With this support, the MDE framework has the potential for increasing the productivity of modelers and the quality of complex simulation experiments, while also catering to the demands of diverse simulation studies, as demonstrated in three case studies revolving around electrically active implants.

Furthermore, the model-driven framework facilitates the automatic reuse of simulation experiments. By specifying experiments in a tool-agnostic manner based on metamodels, the framework allows for easy reuse of experiments from other simulation studies through implemented transformations to and from target specification languages. This capability was demonstrated for the cross-validation with a related cell signaling model.

Finally, the MDE framework offers interfaces through which it can be integrated with other software to fill in the ingredients of the metamodels automatically. This facilitates the (partial) automation of simulation studies as demonstrated by connecting the MDE framework to a workflow system for FEA, which automatically generated and executed convergence tests based on information gathered within the conceptual model.

The presented approach, thus, fulfills the typical expectations and advantages associated with MDE and will be an asset for more effective and systematic simulation studies. The user only needs to fill in the information required by the respective metamodels, obviating the need to

manually write experiment code and acquire expertise in heterogeneous tools and experiment specification languages. Thereby, the approach promises to reduce the effort of conducting simulation experiments and to increase the accessibility of various experiment types. In addition, the experiment metamodels provide guidelines for restructuring existing and implementing new tools, and particularly their interfaces. They therefore promote a more structured approach for developing modeling and simulation software.

The framework may be complemented with an editor for developing new and adapting existing metamodels of the various modeling and simulation approaches as well as advanced experiment types. For this, the composition and reuse of metamodel parts needs to be refined and substantiated with clear semantics. In addition, a web-based repository (including the editor) will be crucial for sharing and collaboratively developing the metamodels and their parts. What should become part of a metamodel needs to be debated in the various modeling and simulation communities, possibly via standardization bodies such as the Simulation Interoperability Standards Organization[36] (SISO). Regarding the automation of simulation studies, further support components can be built on top of the MDE pipeline, such as an automatic selection and parametrization of experiment types and methods (see [238]), or the automatic reuse of previously defined simulation experiments in various contexts (see Chapter 5).

[36] Simulation Interoperability Standards Organization. `https://www.sisostandards.org/`, last accessed 19 July, 2024.

4 Making the Context of a Simulation Study Explicit

4.1 Motivation

"For me context is the key—from that comes the understanding of everything." This quote by American painter Kenneth Noland holds also true in the realm of simulation studies. There, context plays a crucial role in understanding the reasoning behind modeling and analysis decisions, interpreting simulation results, assessing the scope and limitations of a model, and enabling reproducibility and reusability.

With respect to simulation experiments, context influences the specification and execution of experiments. For example, context about the parameters of a model may determine which factors and levels need to be explored in a Latin hypercube experiment design. Likewise, the availability of data from related simulation studies may influence whether cross-validation experiments can be conducted and if so, how a suitable statistical model checking experiment may look like. Furthermore, knowing whether a given model has been calibrated or validated can determine the selection of a particular type of experiment and corresponding methods. Consequently, context is key in conducting simulation experiments—and simulation studies, in general—in a more systematic and effective manner, and doing so automatically.

Context encompasses a variety of knowledge, including the study's goals, the conceptual model describing assumptions or model components, the different stages of the modeling and simulation life cycle already passed, experiment types and methods available or already applied, and the specific tools used. Furthermore, context is provided by the relations to other simulation studies, as well as specifics of the application domain.

Motivated by the reproducibility crisis of simulation, reporting guidelines have increasingly encouraged modelers to make context a central part of a model's documentation. The resulting documentations, however, mostly rely on verbal narratives, which are notoriously ambiguous and therefore difficult to exploit automatically. To enable automatic support for simulation studies, it is crucial to define this knowledge explicitly and make it machine-accessible, also considering how the information relates to each other.

To make context more accessible for automatic support of simulation studies and, in particular, an automatic experiment generator, three research questions can be defined:

1. How can early-stage products of a simulation study be made explicit, focusing on the conceptual model and its artifacts such as research questions, requirements for output behavior, assumptions, simplifications, qualitative models, and data sources?

2. How can the story of a simulation study, including the various model building and analysis activities as well as the products that contributed therein, be made explicit and accessible within and across simulation studies?

3. How can rules about the execution of different types of simulation experiments and their roles in the modeling and simulation life cycle be encoded and linked to contextual information about the simulation study?

The main part of this chapter is structured into two sections, addressing research questions 1 and 2 above. For each question, the state of the art is presented (including revised content

from own research papers [83, 28, 299, 43, 52, 300, 238, 212, 301, 302]), followed by a novel approach contributing to solving the open challenges, and a case study using a prototypical implementation.

The approaches and case studies are based on revised and extended content from published research [83, 302]. The first approach (Section 4.2) introduces a new definition of the conceptual model, seeking a compromise between existing approaches in terms of formality and scope [83, © 2020 IEEE.]. The approach is probed for conceptualizing and documenting a Wnt signaling model at the beginning of, but also throughout, the simulation study. The second approach (Section 4.3) explores the powerful combination of provenance graphs and model databases to make relations between the simulation studies, their products and activities explicit and accessible [302, © 2022 IEEE.]. The BioModels database is used as a case study to showcase how provenance graphs can enrich the information contained in the model database and the conclusions drawn from it.

The discussed approaches later serve as building blocks for a larger framework that enables generating simulation experiments automatically by reuse and adaption (Chapter 5). There, research question 3 from above comes into play, referring to the assembly of these building blocks. Provenance information about model building and previous simulation experiments determines which simulation experiments to generate next. The exact nature of these simulation experiments and the adaptions needed from previous versions are determined by the metadata provided in the provenance, particularly the entities of the conceptual model.

4.2 Making Early-Stage Products Explicit

The conceptual model plays an important role in conducting simulation studies, which is documented in various life cycle models [2, 3]. It provides a way of planning the modeling and analysis steps as well as the simulation study as a whole (e.g., including the research questions to be addressed, collection and preparation of data, and specification of requirements). Thus, it is developed in the early stages of a simulation study before the simulation model is built, or alongside with the model building process. Beyond that, conceptual models provide a documentation of important modeling decisions, including crucial information for reproducing the simulation study, or for enabling reuse of simulation models, simulation experiments, and other products.

In other areas of computer science, and in particular in software engineering, widespread agreement about the definition and specification of conceptual models has been established. There, the conceptual model refers to identifying the software structure, important operations and interfaces, before the technical realization and development of algorithms [303]. Common methods for conceptualization are ER diagrams, Petri nets, state charts, flow charts, or algebras [304]. Some of these approaches are sophisticated, formal modeling languages such as Petri nets, whereas others, such as UML, have a rather ambiguous semantics that is open to interpretation [305]. The decision on which method to use largely depends on the task at hand as well as experience and common practices.

In the modeling and simulation community, the perception of conceptual models varies strongly and with it the possible means for specification. The scope of the conceptual model may refer to either a rather narrow view that defines the conceptual model as an abstract model description [306], or a wider interpretation as loosely-coupled construct that integrates a variety of different communicative artifacts [2], or even broader including the model's context, such as objectives, requirements, and assumptions [307]. Another discussion is how to specify the conceptual model and its parts. There, the ideas range from informal specification using verbal narratives and sketches [75] to formal conceptual modeling languages [258].

As a result of these different views, little practical support for conceptual modeling exists so far. In contrast, in software engineering there is a long-standing research community for conceptual

modeling [308]. The efforts of that community are paying off, e.g., in automatic code generation, or automatic regression testing and consistency checking. For simulation studies, thorough conceptual modeling can also enable automatic support of certain modeling and analysis steps such as reuse and composition of models, or verification and validation [309]. However, "developing an engineering discipline of conceptual modeling [in modeling and simulation] will require much better understanding of: 1. how to make conceptual models explicit and unambiguous [...], 2. the processes of conceptual modeling [..., as well as] 3. architectures and services for building conceptual models" [8].

This section focuses on the first question, i.e., how to make conceptual models, their parts, and the relations of these parts explicit. An approach is taken that aims at integrating the heterogeneous definitions of "conceptual model" in modeling and simulation and the various ways of specification. A broad definition of the conceptual model, i.e., as a loose collection of early-stage products of the simulation study, holds the potential to unify existing definitions, but also poses specific challenges for specification. To approach these challenges, without claiming to be exhaustive, a set of products is identified (which includes, among others, research questions, data, and requirements), and the relations and properties of these products are defined and structured. For the definition of the products and their metadata, formal approaches (DSLs and ontologies) are incorporated where appropriate, while maintaining flexibility and ease of use for the modelers.

The focus will be on a specific type of simulation study, namely those where the building of a valid simulation model is in the center of interest. Consequently, some parts of the conceptual model, such as input data or behavioral requirements, will play a key role in this broader, partly formalized definition of the conceptual model. Based on the Wnt signaling case study and a prototypical wiki implementation, it is shown how the formal and explicit definition of the conceptual model assists in building and analyzing a simulation model. In addition, possibilities for exploiting the gathered information for automatic experiment generation are discussed.

4.2.1 State of the Art

Various approaches exist for making the early stage products of a simulation study explicit. These include conceptual modeling, reporting guidelines, domain-specific languages, and ontologies.

Conceptual Models

During a simulation study, the conceptual model assists the diverse problem analysis, model building, and experimentation activities. This is reflected in life cycle models such as those by Balci [2] and Sargent [3], see Section 2.1. If the goal of the simulation study is to obtain a valid simulation model from which meaningful conclusions can be drawn, these activities are closely intertwined and the conceptual model as well as the simulation model are successively revised, changed, and validated.

However, despite its centrality for the modeling and simulation life cycle, no unanimous definition of the conceptual model has been agreed upon [310]. Figure 4.1 summarizes the commonalities and differences in the definitions of the conceptual model regarding the content of the conceptual model and its specification: whether it is interpreted as having a more narrow or wider scope, and whether it is seen as a formal or informal construct. In the following, the state of the art on conceptual modeling is described in more detail.

An early definition by Nance distinguished between the notions of conceptual model and communicative model [306]. The conceptual model is "[a] model which exists in the mind of the modeler" [306]. This is related to the idea of mental models, i.e., "personal, internal representations of external reality that people use to interact with the world around them" [311]. In contrast, communicative models refer to an (informal) representation of the mental model: "which can be communicated to other humans and can be judged or compared against the system

and the study objectives by more than one human" [306]. Fishwick also sees the conceptual model as vague and ambiguous but techniques, such as semantic networks or databases, may allow some structuring of the knowledge [312].

Other authors (e.g., Cetinkaya et al. [313]) concentrate on formal languages for specifying conceptual models, which may then be automatically transformed into a computerized model (e.g., based on DEVS), thereby blurring the line between conceptual model and computerized model. Guizzardi and Wagner define the conceptual model to be "concerned with identifying, analyzing and describing the essential concepts and constraints of a real-world domain with the help of a (diagrammatic) modeling language [...]" [258]. Their conceptual modeling language OntoUML clearly assigns each model part with an ontological concept, such as entity type, datatype or causal law. Being an extension of UML, automatic translations exist into executable code. CML-DEVS (Conceptual Modeling Language for DEVS), on the other hand, is based on mathematical and logical expressions, which may be translated into the input language of a particular DEVS framework [314].

In addition, template-based frameworks can assist modelers in specifying the components of a (cyber-physical) system [315]. The examples show that these types of conceptual models are focused on making the logic of the simulation model explicit, therefore often focusing on a particular problem domain, and that the expressiveness of the conceptual model is usually constrained by the underlying formalism. Other examples that concentrate on modeling the system structure but are not directly translated into an executalbe simulation model include the Hierarchical Control Conceptual Modeling (HCCM), which was developed to support the specification of control structures [316]. The Activity-Based Conceptual modeling (ABCmod) framework also focuses on modeling structural and behavioral aspects, and provides templates, e.g., for specifying inputs and outputs of the system entities [317].

On the contrary, Balci describes the conceptual model in a broader sense as a "repository of highlevel conceptual constructs and knowledge specified in a variety of communicative forms (e.g. animation, audio, chart, diagram, drawing, equation, graph, image, text, and video) intended to assist in the design of any type of large-scale complex M&S application" [2]. In this definition, the conceptual model refers to aids for constructing the simulation model itself, and is kept separate from artifacts that refer to the context of the simulation model, i.e., the problem formulation, objectives, and requirements. Sargent provides a similar, however shorter, definition: a "mathematical/logical/graphical representation (mimic) of the problem entity developed for a particular study" [3].

In contrast, Robinson subsumes all these different artifacts under the term conceptual model, as all of them, including research questions, requirements, etc., are useful for conducting the simulation study [82, 307]. Robinson also identifies further contents of the conceptual model that should be made explicit: general project objectives regarding, e.g., visualization or simulation speed; model inputs, outputs, as well as used data; scope, level of detail, assumptions, and simplifications; entities, activities, and what modeling approaches to use; and finally justifications for each design choice. Pace identifies a similar list of important conceptual model parts [318]. Both Robinson and Pace recommend that a partly formal approach will be needed, however, neither give concrete suggestions for the notation of conceptual models. The framework is reorganized by Chwif et al., who define four parts of the conceptual model: objectives, complexity, input and output, and scenarios [319].

Building on those definitions, Fujimoto et al. define the conceptual model as a collection of early-stage products that integrate and provide information and requirements for a variety of simulation study aspects [8]. They also stress the need for developing more explicit and formal conceptual models based on domain-specific languages, ontologies, and other suitable knowledge representations. However, they do not provide solutions on how to do so. Furthermore, although many formalisms exist, they are not accessible to domain experts and integration into the usual workflows is difficult. An exception is, e.g., the artifact-based workflow by Ruscheinski

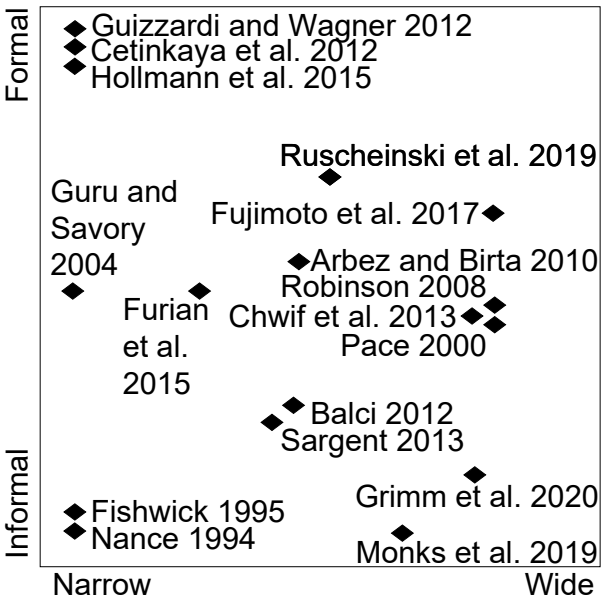


Figure 4.1: The spectrum of conceptual model definitions from narrow to wide and informal to formal. Extended from Wilsdorf et al. [83].

et al. where some of the discussed conceptual aspects have been integrated with special focus on requirements formally defined as temporal logic formulas, thereby making this part of the conceptual model formally explicit and exploitable [223]. However, it is a heavyweight approach, which also states something about the process of designing a conceptual model and how to store it.

The work on reporting guidelines for simulation models faces similar challenges to those stated by Fujimoto et al. The ODD (Overview, Design concepts and Details) protocol by Grimm et al., for example, has recently been extended to the context of simulation studies and now includes purpose and requirements (called "patterns") of the model, state variables, processes, design concepts, initialization, input data, and submodels [75]. It provides a general document structure, however, no detail about how the individual elements should be specified. Similarly, the reporting guideline STRESS (Strengthening the Reporting of Empirical Simulation Studies) by Monks et al. provides a checklist addressing the documentation of objectives, model logic, and data; however, in an informal manner [246]. Like ODD, STRESS documents are typically created retrospectively to summarize the work done in a mainly verbal format, whereas the activity of conceptual modeling typically accompanies the whole modeling process.

Looking at the published definitions of the conceptual model that are compared in Figure 4.1, narrow definitions of the conceptual model use either formal or informal approaches exclusively. Wider definitions, on the other hand, use a variety of specification formats, and thus are often semi-formal. However, there is a lack of *wide and formal* approaches. This section is aimed at a lightweight approach for integrating the diverse parts of a conceptual model using formal methods where applicable. The approach will treat conceptual modeling in the broader sense including requirements engineering and project documentation in addition to describing the real-world problem domain and identifying the relevant types of objects and events, and how they should be modeled.

Reporting Guidelines

There exists a variety of guidelines for describing and documenting simulation models. Their goal is to provide modelers with a de facto standardized structure to consistently describe the model's contents and purpose in a format that is readable and understandable by others. In

addition, reporting guidelines may include the documentation of various conceptual artifacts, such as requirements, assumptions, qualitative model, and data. Depending on the reporting guidelines at hand, a more or less broad definition of what needs to be included as context of the model is adopted.

The ODD (Overview, Design concepts and Details) protocol, for example, includes purpose and requirements (called "patterns") of the model, entities and state variables, processes, design concepts, initialization, input data, and submodels [75].

STRESS (Strengthening the Reporting of Empirical Simulation Studies) emphasizes what a simulation study is about in terms of objective, (model) logic, data, experimentation, implementation, and code access [246].

MIRIAM (Minimum Information Required in the Annotation of Biochemical Models) provides an annotation scheme that allows linking the model to external knowledge (e.g., various domain ontologies) [320]. Semantically enriched models according to MIRIAM may, e.g., unambiguously specify what entities are included in the model (via the Gene Ontology [321]), which cell line the model refers to (via the Cell Line Ontology [322]), or whether it is implemented as a rule-based model or using process algebra (via the Mathematical Modelling Ontology [323]).

TRACE (TRAnsparent and Comprehensive model Evaludation) is designed to capture a simulation study as a whole [236]. It thus explicitly contains sections pertaining to early-stage products, i.e., problem formulation, data evaluation, and conceptual model evaluation, as summarized in Table 4.1. The latter should specify "[t]he simplifying assumptions underlying a model's design, both with regard to empirical knowledge and general, basic principles" with the aim to "[allow] model users to understand that model design was not ad hoc but based on carefully scrutinized considerations." In addition to that, TRACE uses ODD for model description (see Table 4.1, Element 2). The remaining elements of TRACE are presented in Table 4.2 and discussed in Section 4.3.1, which explores the relations between products, activities and studies.

Reporting guidelines also consider domain-specific aspects of the M&S products. MMRR (Minimum Model Reporting Requirements) and PMRR (Preferred Model Reporting Requirements) are protocols in systems dynamics [245], and thus include "equations and algorithmic rules, all model parameters and initial values", and a definition of all variables with units of measurement as well as qualitative and quantitative data sources from which equations were derived. In the context of agent-based modeling, ODD [75] suggests describing all agents and their environment as well as temporal and spatial resolution. RAT-RS (Rigour And Transparency – Reporting Standard) is a documentation standard for data use and data requirements in agent-based modeling, which includes opinions and uncertainties of agents [324]. The reporting guidelines by Erdemir et al. [62] target finite element analysis simulation studies, and therefore request information with respect to boundary conditions, the mesh, and convergence criteria for the numerical simulation.

However, ODD and the other reporting guidelines have their challenges [75]. In particular, guidelines are of voluntary nature, and it is difficult to ensure that they are followed by the individual modeler or the modeling community. Still, with increasing awareness of the FAIR (Findable, Accessible, Interoperable, Reusable) principles [325], some model repositories and scientific journals now encourage modelers to provide a documentation of their simulation studies in accordance with community-defined guidelines, for instance, the CoMSES[37] (Computational Modeling in the Social and Ecological Sciences) repository for agent-based models and the Journal of Artificial Societies and Social Simulation (JASSS)[38]. Moreover, reporting standards leave it open how to represent the requested information. They provide a rather general checklist or document template. Therefore, it is up to the modeler to interpret and implement the guidelines. Furthermore, reporting documents are often created retrospectively to summarize the work

[37] `https://www.comses.net/`, last accessed 19 July, 2024.
[38] `https://www.jasss.org`, last accessed 19 July, 2024.

Table 4.1: Structure and contents of TRACE documents (elements 1–4), verbatim from Grimm et al. [236]. For elements 5–8, see Table 4.2.

TRACE element	This TRACE element provides supporting information on:
1. Problem formulation	The decision-making context in which the model will be used; the types of model clients or stakeholders addressed; a precise specification of the question(s) that should be answered with the model, including a specification of necessary model outputs; and a statement of the domain of applicability of the model, including the extent of acceptable extrapolations.
2. Model description	The model, i.e. a detailed written model description. For individual/agent-based and other simulation models, the ODD protocol is recommended as standard format. For complex submodels, include concise explanations of the underlying rationale. Model users should learn what the model is, how it works, and what guided its design.
3. Data evaluation	The quality and sources of numerical and qualitative data used to parameterize the model, both directly and inversely via calibration, and of the observed patterns that were used to design the overall model structure. This critical evaluation will allow model users to assess the scope and the uncertainty of the data and knowledge on which the model is based.
4. Conceptual model evaluation	The simplifying assumptions underlying a model's design, both with regard to empirical knowledge and general, basic principles. This critical evaluation allows model users to understand that model design was not ad hoc but based on carefully scrutinized considerations.

done. Their purpose is therefore different from that of the conceptual model, since conceptual modeling is an activity that is meant to accompany the whole modeling process, especially in the early phases but also throughout. Still, the work on reporting guidelines can inspire how the conceptual model needs to look like to capture all important aspects about a simulation model that allow facilitating (prospective) guidance of a modeler in building, analyzing, or reusing a model and the study results. Such (semi-)automated guidance is also facilitated by (semi-)formal representations of the conceptual model. Reporting documents, on the other hand, usually lack formal specifications.

To alleviate said challenges, a more formal and rigorous approach to model and context documentation is required, as well as tools that support modelers in creating these kinds of documentations. Solutions, thus far, include the documentation via so-called modeling notebooks [326]. For instance, Jupyter Notebook can ease the burden of documenting simulation studies and enrich the documentation with interactive, computational elements. Moreover, semantics can be assigned to the given information by using ontologies. This would, for instance, enable automatically reusing conceptual aspects during the simulation study, as shown in the context of Jupyter notebooks [252].

Domain-specific Languages

For the formalization of single early-stage products, domain-specific languages (DSLs) play a crucial role. A domain-specific language "provides a notation tailored towards an application domain and is based on the relevant concepts and features of that domain" [327]. DSLs are designed towards "improving productivity for developers and improving communication between domain experts" [328]. Unlike general-purpose programming languages they take the needs of a specific domain into consideration and focus on ease-of-use as well as conciseness of the specifications.

A DSL typically consists of three parts: the abstract syntax defines the high-level structure of the language, the concrete syntax describes how code in the DSL looks like, and the semantics refers to the meaning of the language's constructs. There are internal DSLs, which are embedded in a

host language and thus enjoy the flexibility of the Turing-complete general purpose programming language [328]. On the other hand, they may lack conciseness and ease-of-use. Most DSLs, therefore, are realized as external DSLs. This means that they come with an own abstract syntax and parser. They, consequently, tend to be more expressive than their internal counterpart, better accommodate the domain's needs, and have tailored, highly-efficient engines for their evaluation. On the downside, the more the language is extended by domain-specific constructs, its advantages in terms of productivity and communication tend to fade [328].

A myriad of DSLs have been proposed for specifying behavioral requirements of the simulation model. Temporal logics, such as Linear Temporal Logic [329], describe the simulation output trajectories in terms of the operators *Globally* (property holds at all time instances), *Eventually* (property holds some time in the future), and *Until* (describing a sequence of two properties). Interval temporal logics, such as Metric Interval Temporal Logic (MITL) [167] or Signal Temporal Logic (STL) [168], allow assigning time intervals to the temporal operators. Also various extensions exist for these temporal logics. Signal Spatio-Temporal Logic (SSTL), for instance, adds a number of spatial operators that allow expressing spatial patterns in the model output over time [170].

Other languages for requirement specification are custom-built to feel natural to a user and domain-expert. The hypothesis language FITS (Formulating, Integrating and Testing of hypotheses in computer Simulation), e.g., provides means for specifying hypothesis tests [330]. Another example is the Biology Property Specification Language (BPSL), which allows encoding quantitative and qualitative experimental data as simple inequality constraints on the model output [138]. In addition, the challenge of making stylized facts (i.e., simplified statements about empirical data) formally explicit has been discussed in [97], and a first language draft for combining temporal logics with statistical tests on time series was presented.

DSLs can also be used for specifying a first conceptual representation of the model to be built. Here, in particular, DSLs that have a graphical representation are suited for conceptual modeling and specifying the qualitative model. External DSLs with a graphical notation include the Unified Modeling Language (UML) [271], the Systems Modeling Language (SysML) [331], Modelica [332], and BPMN [333]. They provide means for specifying systems in terms of their (hierarchies of) objects, processes and interactions of those objects. In addition, UML, as well as schema languages (e.g., JSON Schema, YAML Schema or XML Schema), present so-called metamodeling languages and are crucial in creating new domain-specific languages. An example of this was shown in Chapter 3, where a domain-specific language for simulation experiments with a metalayer was created. In the spirit of UML, which subsumes multiple diagram types, graphical DSLs that may be used for qualitative modeling in a particular domain can be developed as community-based projects. An example of this is the Systems Biology Graphical Notation (SBGN) [334], a standard for visual representation of biological networks, which includes a process diagram, an entity relationship diagram, and an activity flow diagram.

To summarize, for the formal specification of requirements as well as model components and logic various options exist. However, extensions may be required that tailor the generic temporal logics or modeling formalisms to a domain. Furthermore, additional work is needed especially towards DSLs for specifying assumptions and simplifications.

Ontologies

As mentioned above in the context of reporting guidelines, ontologies present an important means for making information about early-stage products explicit and unambiguous as well as findable and accessible. They provide a controlled vocabulary and semantics [335] that, e.g., enables reuse and interoperability of model components [265].

Ontologies are defined as "a formal, explicit specification of a shared conceptualisation" [336]. According to this definition, DBPedia, for instance, clearly can be considered an ontology as it provides structured, semantically enriched information extracted from Wikipedia [337].

In addition, DBPedia is a machine-readable knowledge graph formalized via RDF (Resource Description Framework), and it has been the result of a massive community project.

Ontologies can also be tailored to assist modeling and simulation in a specific domain. They also may be customized towards specifying a particular type of artifact of the simulation study. Most ontologies that exist so far in the realm of modeling and simulation are designed and used for specifying the model content (referring to the qualitative model description) as well as clarifying what the research questions are about. In the area of systems biology, a number of such ontologies have been created, including the Cell Line Ontology [322], the Pathway Ontology [338], the BRENDA Tissue Ontology (BTO) [339], or the Gene Ontology (GO) [321]. In physics and multiphysics simulation, the Physics-based Simulation Ontology (PSO) organizes concepts such as material properties, physical properties, or shape [267]. The Simulation Intent Ontology also organizes different attributes of the model and the simulation but, in addition, includes the simulation objectives and requirements [340].

Behavioral requirements regarding the model output may be annotated using the various ontologies mentioned above to unambiguously specify the output variables. Similarly, data sets may be annotated with those ontologies to specify their content. Referring to model assumptions, additionally, the Systems Biology Ontology (SBO) specifies, e.g., concepts with respect to the assumed kinetic rate laws or assumed role of an entity [335].

Moreover, unambiguously specifying what methodology was used is crucial for reproducing and reusing artifacts, as already emphasized by the ontology of tasks and methods [341]. This includes making information explicit about the modeling approache(s), or what methods were used for the simulation experiments. The former can be specified using SBO, which contains a selection of discrete, continuous, or hybrid approaches, or using the ontology for Discrete-event MOdeling and simulation (DeMO) when modeling discrete-event systems [266]. For the latter, rudimentary ontologies for sensitivity analysis [238] and a taxonomy for simulation-based optimization have been developed [155]. Furthermore, the factors to be varied in a simulation experiment may be specified using the Experimental Factor Ontology [342].

However, work needs to be done in developing ontologies for all artifacts of the simulation study and their metadata. Moreover, existing ontologies need to be aligned, extended and combined to allow for automatic interpretation, generation and adaption of simulation experiments and other artifacts. E.g., in the case of ontologies for units of measurement, different levels of ontologies exist [343]: 'Ontologies of the first level "standardize characters for unit symbols and operators". On the second level they "additionally represent the concepts of unit and dimension" and "support the operations of unit conversion and unit conformance". On the third level they "additionally include the concept of quantity and system of quantities". On the fourth level they additionally provide a "full descriptions of quantities, prefixes, ordinal and nominal measurement scales, and systems of quantities and units". On the fifth level they "also incorporate the concepts (precision, accuracy, distribution etc.), relations and operations associated with calculating and representing measurement uncertainty"' [343]. Depending on what needs to be expressed explicitly and what shall be supported automatically, a different level of ontology is necessary.

The development of an ontology involves defining classes in the ontology, arranging the classes in a taxonomic (subclass–superclass) hierarchy, defining attributes and describing allowed values for these attributes, and filling in attribute values for class instances [344]. Moreover, it usually involves mapping to a top-level ontology, such as the Basic Formal Ontology (BFO), and encoding in a format such as RDF (Resource Description Framework) or OWL (Web Ontology Language). Developing an ontology, therefore, is a challenging and time-consuming task. Increasingly, natural language processing and machine learning may move the field of ontology development in modeling and simulation forward [345]. Advancements in automatic literature sweeps, extraction of core concepts and inferring their relations will become crucial in ontology construction, as well as the implementation of these functionalities in a ready-made tool. But in addition to the challenge of collecting and organizing all this knowledge, various other issues need to be addressed. For

instance, methods need to be developed that will actually use these ontologies, relate ontologies to each other, and do this automatically [264].

4.2.2 Case Study

A conceptual model is built for the simulation study of Wnt signaling in human neuronal progenitor cells during early differentiation, following the storyline of Haack et al. (2015) [31]. The case study shows how the conceptual model can assist in developing the simulation model and implementing it in ML-Rules.

Furthermore, it shows how the conceptual model assists in conducting simulation experiments (using SESSL). In particular, based on the conceptual model, a validation experiment can be generated automatically. In this validation, the model output is compared against wet-lab data in a statistical model checking.

In the validation results, it is observed that the given qualitative model and the assumptions do not suffice to reproduce the experimental results. In detail, the β-catenin concentration is increased at certain time points despite the disruption of the lipid rafts, which means that the requirement associated with the research objective cannot be met by the simulation model.

This indicates necessary revisions in the conceptual model, e.g., by further experimental measurements, additional literature research, and possibly the definition of additional requirements. Indeed, in the follow-up study by Peng et al. [228] it is shown how the simulation model was successively composed to resolve the problem, and finally validate the model.

The following section discusses the explicit representation of the conceptual artifacts, e.g., the requirements as temporal logic formulas, and what role this plays in generating the said validation experiment.

4.2.3 An Integrated Conceptual Model Definition

The aim of this section is to integrate all the different conceptual aspects and to make them explicit into a data structure. At the same time, the developed approach should support both formal and informal specifications, as both are crucial for the conceptual modeling phase and can serve different purposes when defining the different early-stage artifacts.

The proposed definition of the conceptual model encompasses the main artifacts research question *rq*, requirement *r*, qualitative model *qm*, assumption *a*, methodology *m*, and data or information source *src*. Using an EBNF-style notation [270], the conceptual model *cm* is defined as a tuple that comprises lists of zero, one, or many (denoted by *) of the different artifacts:

$$cm = \left(rq^*, r^*, qm^*, a^*, m^*, src^*\right)$$

In the following, the artifacts are defined and for each, examples from the case study are discussed.

Research Question

A simulation model is built for a system and some experiments to answer specific questions about the system [15]. These questions (or research objectives) determine when a suitable abstraction of the system of interest has been achieved. The research question *rq* is defined as a tuple that may integrate several objects for description (d) of the overall objective and its context. This, typically informal, specification of the research objective may be further substantiated by specific subgoals, called requirements. Thus, the research question may contain links (denoted by @) to requirement artifacts *r* that express precisely what observed phenomena shall be reproduced by the simulation model. There may be a list of zero, one, or many related requirement artifacts (denoted by *). This approach is prominent in areas where the simulation models are built for explanation rather than prediction. In addition, the research question is assigned an identifier (id) represented by a unique string. Identifiers are used to distinguish pieces of information in

the conceptual model. They are therefore required (denoted by !) when specifying an artifact.

$$rq = \big(id!, d^*, @r^*\big)$$
$$id = unique\ text$$

A description d is defined as multimedia object consisting of a type *dType* (the alternatives such as 'text', 'image', or 'video' are terminal symbols of the definition and are delimited by |), as well as a *format* and a *tool* for handling a description of that type given as text. And finally, it contains the actual specification (*spec*), which may be given either by text or by a reference to a file. A special case is the description via the type 'ontology'. This allows linking the artifact to an external knowledge base via an ontology tag. The information fields *format*, *tool*, and *spec* (specification) will also be used in the definitions of the other artifacts. Moreover, each artifact will have a type; however, the type options will differ between the artifacts. All these aforementioned properties, when instantiating the definition, can be specified zero or one times. Furthermore, links to *src* artifacts allow setting artifacts into context, e.g., if the research question of the current study builds on previous work. The description will be part of all the different ingredients of the conceptual model, and thus add flexibility by allowing the use of different communicative forms as proposed in Balci's definition of the conceptual model [2].

$$\begin{aligned} d &= \big(dType, format, tool, spec, @src^*\big) \\ dType &= \text{'text'} \mid \text{'image'} \mid \text{'video'} \mid \text{'ontology'} \mid \ldots \mid \text{'other'} \\ format &= text \\ tool &= text \\ spec &= text \mid @src \end{aligned}$$

Case Study: At the beginning of the Wnt simulation study, the focus is laid on membrane dynamics, and lipid rafts in particular. Lipid rafts are small domains in the membrane (microdomains) with high local concentration of cholesterol, sphingolipids, and protein receptors. Therefore, the research objective is defined as: "What is the role of membrane lipid rafts on canonical Wnt signaling in human neural progenitor cells?" Instantiating the definition of the research question (rq), the data structure of this artifact looks like this:

(id = Main Objective,
d = [(dType = text,
spec = What is the role of membrane lipid rafts on canonical Wnt signaling in human neural progenitor cells?)],
r = [*@Req1*])

Since a research question is typically not given formally, it will not directly be used for automatic experiment generation (but the associated requirements will). However, the research question is relevant for interpreting the results of the validation and conducting the next model building steps. In particular, following the failed validation experiment, additional information and data about LRP6 and its interaction within the Wnt signaling pathway need to be collected.

Requirement

Requirements usually refer to the output and thus the behavior of the simulation model. If the expected behavior is expressed formally as a logic formula (e.g., the temporal logic MITL [167]), it can be checked automatically via statistical model checking [159] in a tool like SESSL [36]. The definition of a requirement artifact therefore contains, in addition to an identifier and a multimedia description as defined above, information about the temporal logic used, i.e., the type of logic, a tool for interpreting the logic specification, and the actual specification of the formula. Note, that no specification format may be necessary, since the syntax used to express temporal logic formulas typically depends on the tool at hand. Therefore, often format and tool will be redundant information, and providing only the name and version of the tool would suffice.

Having exact information about the tool used for specifying and evaluating a temporal logic formula is also crucial for being able to reproduce the simulation study, since different model checkers may use different interpretations of the same formula [97].

Another type of behavioral requirement is to reproduce data, i.e., the requirement is directly linked to a data source (*src*) that describes the intended behavior. The check, whether the output of the simulation model is sufficiently close to the expectation, is often done by applying face validation or by calculating the mean squared error.

In addition to expectations regarding the model's behavior, requirements might also be more fundamental and refer to non-functional properties of the simulation model and its implementation. They may refer, e.g., to the performance of the simulations, or to the choice of the modeling and simulation approach [307]. For example, if spatial resolution plays a role, the modeling and simulation approach should take space into account [346]. If small numbers of entities need to be considered, a stochastic approach based on the Gillespie algorithm [37] might be more suitable than a deterministic one. Also, for some simulation studies the methodology is determined at the project start due to other constraints, and may be represented explicitly by a methodology artifact *m*.

Furthermore, to create a knowledge graph that allows tracing the origin as well as the usages of an artifact, a requirement can be linked to the research questions it refers to. In many simulation studies there will be only one objective, however, for larger projects multiple independent objectives could be formulated.

In summary, the definition of the requirement artifact looks like this:

$$
\begin{aligned}
r &= \big(id!, d^*, rType, @rq^*\big) \\
rType &= behavioralR \mid @m \mid \ldots \\
behavioralR &= temporalLogic \mid @src \mid \ldots \\
temporalLogic &= \big(format, tool, spec\big)
\end{aligned}
$$

Case Study: To refine the rather general research question, a first behavioral requirement (Req1) of the membrane processes can be defined. For instance, lipid rafts typically cover $25-30\%$ of the membrane, and therefore the concentration of homogeneously distributed LRP6 (Low-density lipoprotein Receptor-related Protein 6) within lipid rafts should stay between 25% and 30% of total LRP6 within any given time interval, whereas the concentration of the membrane-associated protein CK1γ should stay between 75% and 85%. This requirement can be specified in the time interval of 60 to 720 minutes using Metrical Interval Temporal Logic (MITL) [167] as supported by the statistical model checking module of SESSL [36]:

$$
\begin{aligned}
\phi_1 = {} & \texttt{G}(60,720)((\texttt{OutVar}(\texttt{d\_lrp6}) >= \texttt{OutVar}(\texttt{lrp6}) * \texttt{Constant}(0.25)) \texttt{ and} \\
& (\texttt{OutVar}(\texttt{d\_lrp6}) <= \texttt{OutVar}(\texttt{lrp6}) * \texttt{Constant}(0.3))) \\
\phi_2 = {} & \texttt{G}(60,720)((\texttt{OutVar}(\texttt{d\_ck1y}) >= \texttt{OutVar}(\texttt{ck1y}) * \texttt{Constant}(0.75)) \texttt{ and} \\
& (\texttt{OutVar}(\texttt{d\_ck1y}) <= \texttt{OutVar}(\texttt{ck1y}) * \texttt{Constant}(0.85)))
\end{aligned}
$$

After adding a textual description as well as an information source, the definition of the requirement artifact is given by the following tuple:

$$
\begin{aligned}
\big(\text{id} &= \text{Req1}, \\
\text{d} &= [(\text{dType} = \text{text}, \\
&\quad\ \text{spec} = \text{Between 25 and 30\% of total LRP6 should reside within lipid rafts, as well as 75-85\% of total CK1}\gamma., \\
&\quad\ \text{src} = [@\textit{Sakane et al. (2010)}])], \\
\text{temporalLogic} &= (\text{format} = \text{MITL}, \\
&\quad\ \text{tool} = \text{SESSL}, \\
&\quad\ \text{spec} = \phi_1 \wedge \phi_2), \\
\text{rq} &= [@\textit{Main Objective}]\big)
\end{aligned}
$$

Based on this requirement artifact, a statistical model checking experiment can be set up. The temporal logical formula is inserted as the model property to be checked. In the simulation observation, LRP6 and CK1γare added as observed species, until the time $t = 720$. Additionally, as SESSL is given specifically as a tool, the concrete syntax for experiment generation is known.

Qualitative Model

The qualitative model is what nearly all literature about conceptual models expect from it: a list of variables and the causal relations between those. The qualitative model *qm* could be provided simply as a sketch, in a simple rule-based formalism, or as a Boolean model using suitable file formats and tools. In addition, the definition below provides a means to make the important parts of the qualitative model such as entities, their attributes, and activities they participate in explicit. This allows us to reference model structures in other parts of the conceptual model, e.g., as part of assumptions and methodologies, or map them to a specific data source. The definition of the *qm* follows the epistemological hierarchy of systems [347]:

$$
\begin{aligned}
qm &= \big(id!, d^*, qmType, format, tool, spec, @qm^*, @ent^*, @act^*, \\
&\qquad @param^*, @a^*, @m^*, @src^*\big) \\
qmType &= \text{'text'} \mid \text{'sketch'} \mid \text{'rule-based model'} \mid \text{'boolean model'} \mid \ldots \mid \text{'other'} \\
ent &= \big(id!, d^*, value, unit, @att^*, @ent^*, @act^*, @m^*, @a^*, @src^*\big) \\
att &= \big(id!, d^*, attType, @m^*, @a^*, @src^*\big) \\
attType &= text \\
act &= \big(id!, d^*, @ent^*, @param^*, @m^*, @a^*, @src^*\big) \\
param &= \big(id!, d^*, value, unit, @act^*, @m^*, @a^*, @src^*\big) \\
value &= text \\
unit &= text
\end{aligned}
$$

In particular, the *qm* may be structured into a set of submodels, which is essential, e.g., for multilevel [35] or multiscale modeling [348]. Each submodel is also a qualitative model (*qm*), i.e., a recursive definition is allowed. Entities *ent* are specified such that they may be either simple variables (i.e., able to hold a single value), or structured entities that may contain subentities and have attributes *att* (to be used in a compartment- or agent-based modeling metaphor), and each attribute having a type (usually one of the standard data types, such as integer, double or Boolean). Thus, an entity is characterized by an identifier, a description, a value and unit for the initial concentration, a list of attributes, and a potential list of subentities. Activities *act* describe how the entities interact and thus the dynamics of a system. An activity refers to one or more entities, and model parameters (*param*) such as rate constants. When specifying the parameters, like with the entities, we allow the inclusion of quantitative information, represented by a value and a unit, since input of the model is considered a crucial part of the conceptual model [27]. Thereby the definition can bridge between the qualitative model and, e.g., parameter tables collected as part of the data sources.

For the qualitative model, also a format or tool can be chosen in case a more complex formalism is needed that, e.g., supports the specification of the interaction type, attribute types, stoichiometric matrices, or to make the initial states explicit. Again, the specification may be included directly within the qualitative model, or a reference to a file (*src*) may be provided. In addition, the various parts can be linked to assumptions to express additional constraints when building the model, or to methodologies to decide how these model parts should be specified.

Case Study: Based on the defined research question and requirement, a diagram identifying the main structures and reactions of the Wnt model is sketched. The diagram is depicted in

Figure 1.2 of Chapter 1. A reference to this figure (WntModel.png) is stored in a qualitative model artifact, where also the names of submodels, entities and parameters are made explicit:

(id = QM1,
d = [(dType = text,
spec = Diagram of the submodels, entities, and reactions)],
qmType = sketch,
format = PNG,
tool = –,
spec = WntModel.png,
qm = [*@Membrane Model*, *@Axin/β-cat Model*],
ent = [*@LRP6*, *@LRP*, *@CK1γ*, *@Beta-cat*, ...],
act = [*@R1*, *@R2*, *@R3*, ...],
param = [*@W*, ...],
a = [],
m = [],
src = [])

Further information about the components of the model can be made explicit, including the corresponding, attributes, reactions and information sources of each entity. For example, consider the entity LRP6, which possesses a phosphorylation site and participates in two biochemical reactions:

(id = LRP6,
d = [(dType = ontology,
spec = UniProt:O75581)],
value = 4000,
unit = –,
att = [*@Phos*],
ent = [],
act = [*@R1*, *@R2*],
m = [],
a = [],
src = [*@Sakane et al. (2010)*, *@Bafico et al. (2001)*])

It is described by an entry in the UniProt database, and further literature can be found in the referenced publications. Literature and other sources that the model components are based on can be specified within the information source artifacts (see below).

For generating simulation experiments, in particular, the parameter values and initial concentrations of variables specified in the qualitative model are of interest. For instance, in the validation experiment, a line is generated that sets the initial value of LRP6 to 4000.

Assumption

While requirements look at the expectations we have referring to the simulation study, and thus typically focus on the output, assumptions describe the starting point of a simulation study, its scope, and simplifications that are made. Assumptions therefore determine how the simulation results might be interpreted. Assumptions *a* can be stated more generally in the form of text, or specifically using mathematical formulas, which could be provided in a specific format and interpreted by a specific tool. The description of an assumption artifact can provide further meaning and could include complete information about how the assumption was (formally) derived. Assumptions may be derived from data or literature, and thus links to data and information source artifacts can be included.

The formulated assumptions may be related directly to individual entities and causal relations of the system of interest. Therefore, the definition integrates links to the various parts of the qualitative model specification. For example, in a chemical model one could assume that the overall number of phosphorylated entities remains constant during the simulation, or one could make an assumption about the distribution of a parameter or attribute.

The definition of the assumption artifact looks as follows:

$$a = \left(id!, d^*, aType, format, tool, spec, @src^*, @qm^*, @ent^*, @att^*, @act^*, @param^*\right)$$
$$aType = \text{'text'} \mid \text{'formula'} \mid \ldots \mid \text{'other'}$$

Simplifications can be specified similarly to an assumption, and thus will be included in the above definition. However, they serve different purposes in the simulation study: Whereas assumptions are about filling in gaps or uncertainties in the knowledge about the world, simplifications are a choice to model the world more simply to improve transparency and rapid development and execution of the model [27]. For instance, simplifications often refer to the aggregation or omission of model elements. Therefore, they may be treated as individual artifacts.

Case Study: As the Wnt model is an extension of the model by Lee et al. [32], it inherits most assumptions and simplifications from this earlier model. For example, Lee et al. (2003) assumed that the autocrine Wnt signal W can be modeled as a negative exponential process. This statement can be expressed formally, and included in the following assumption artifact, together with some metadata about the publication and the related model parameter:

$$\begin{aligned}
&\big(\text{id} &&= \text{A1},\\
&\text{d} &&= [(\text{dType} = \text{text},\\
&&&\quad \text{spec} = \text{Autocrine Wnt signal modeled as negative exponential process})],\\
&\text{aType} &&= \text{formula},\\
&\text{format} &&= \text{–},\\
&\text{tool} &&= \text{–},\\
&\text{spec} &&= W = \begin{cases} 0 & , \text{ for } t < t_0 \\ e^{-\lambda(t-t_0)} & , \text{ for } t \geq t_0 \end{cases},\\
&\text{src} &&= [@\textit{Lee et al. (2003)}],\\
&\text{qm} &&= [\,],\\
&\text{ent} &&= [\,],\\
&\text{att} &&= [\,],\\
&\text{act} &&= [\,],\\
&\text{param} &&= [@W]\big)
\end{aligned}$$

This particular assumption refers to the model logic (the dynamics of W). Thus, the assumption is used immediately during model building, and is not needed for generating the statistical model checking experiment.

Methodology

Methodology artifacts may refer to different types of methodologies. One type of methodology is the modeling metaphor, such as agent-based or reaction-based. These kinds of artifacts refer to the syntax of the model [349]. They may also refer to the execution semantics that are deemed most adequate to capture the dynamics of the system such as discrete-event-based, discrete-stepwise, continuous or hybrid simulation [350]. If the system at hand exhibits spatial behaviors, a suitable spatial modeling approach has to be selected. Sometimes, also decisions about specification formats or the modeling tools, like NetLogo for agent-based modeling and simulation [282], have to be made at the conceptual level. Similarly, analysis approaches such

as experiment designs [120] have to be carefully selected beforehand. This is important also for simulation studies that use models for prediction.

In a hybrid modeling and simulation setting it is especially important to make explicit which parts of the simulation model each methodology applies to. Therefore, the definition below includes links to the different parts of the qualitative model, i.e., the submodels (*qm*), entities (*ent*), attributes (*att*), activities (*act*), and parameters (*param*).

And again, a description can be added to explicate and justify the choice of methodology as well as the rationale behind it:

$$m = \left(id!, d^*, mType, spec, @r, @qm^*, @ent^*, @att^*, @act^*, @param^*, m^*\right)$$
$$mType = \text{'metaphor'} \mid \text{'dynamics'} \mid \text{'spatial'} \mid \text{'format'} \mid \text{'tool'} \mid \text{'analysis'} \mid \text{'traceability'} \mid \ldots \mid \text{'other'}$$

Since the reproducibility crisis of science, thinking about how to make the own research reproducible and well-documented has become an important task in the modeling and simulation community [351]. Therefore, approaches used for traceability should be selected at the beginning of a simulation study to allow a documentation of the entire model building and experimentation process. Hence, traceability is another type of methodology in the above definition. Approaches for traceability include reporting guidelines like STRESS [246] and ODD [75], scientific- [352] and artifact-based workflows [223], and provenance [28].

Some specified methodologies are related to requirements, i.e., they are mandatory with respect to a research objective or other project requirements. Those requirements may refer to the semantics of the simulation approach, e.g., that a continuous-time Markov chain is required. This still leaves the decision open, e.g., whether to apply a stochastic process algebra or a stochastic Petri net for modeling. In return, if a specific format or tool is requested, still different formalisms may be used. Therefore, the definition of a methodology artifact may relate to other methodology artifacts, making conrete, e.g., what the particular combination of modeling metaphor and tool will be for this simulation study.

To support the choice of modeling approaches, formats, tools, etc., an ontological classification will be required that structures the general concepts of modeling and simulation [258]. Instead of a textual specification of the methodology, a reference to an ontology term may be used. Currently, for instance, the ontologies KiSAO for simulation algorithms [335], and DeMO for discrete-event modeling concepts [266] exist. Uniquely identifying each concept with an ontology tag would facilitate an automatic exploitation of the conceptual model, as well as exploitation of the simulation models and experiments that were based on it.

Case Study: The choice of modeling language and the corresponding modeling tool is closely intertwined with the development of the qualitative model. From the qualitative model of the Wnt model, it becomes clear that a hierarchical, attributed modeling approach should be used. In particular, nested entities are needed to represent the structure of the cell and the separation of the membrane into individual lipid rafts; attributes are needed to account for the various binding and activation (phosphorylation) states of proteins, such as the receptor LRP6. The rule-based modeling language ML-Rules supports all the desired features [35]. In a methodology artifact, the choice can be made explicit:

(id = M1,
d = [(dType = text,
spec = A modeling language is required that allows using nested entities with attributes. Ideally, a rule-based representation should be supported.)],
mType = tool,
spec = ML-Rules,
r = –,
qm = [],
ent = [],
att = [],
act = [],
param = [],
m = [])

The information provided in this methodology artifact is crucial for the automatic generation and execution of the experiment, ensuring that the correct simulator is chosen for running the model.

Data and Information Source

Data plays a central role for simulation studies. Data can comprise input data, i.e., referring to the input parameters of the model such as rate constants, initial concentrations, etc. It can be used for validation or calibration of the simulation model, to establish a theory, or for illustration of a problem or process.

Data may be experimental data, i.e., obtained from real measurements, or data generated by other simulations, used to support calibration or (cross-)validation.

But data that are helpful in conducting the simulation study may also be of a more general nature, and thus called information source (*src*). These can reference, e.g., scientific literature or scientific notebooks [251]. The description may provide additional information about the source, e.g., why a publication was selected, or how a data set was created.

As seen in the above definitions, data and information sources can be linked to the other parts of the conceptual model, e.g., a methodology, an assumption, or specific entities or parameters of the qualitative model. The sources thereby provide meaning to these other artifacts as outlined in the previous subsections. In particular, each source artifact can be resolved to a unique source ID such as a DOI or URL. Alternatively, a local file can be uploaded, e.g., if data have just been produced in own wetlab experiments, and therefore have not been published yet.

To support the analysis or display of the data and information sources, the specification of an appropriate file format or tool is also included. The choice of an appropriate tool for a specific file format may again be further supported by ontologies and other knowledge bases. However, the formatting of data and information is a research topic of its own, e.g., how to express observational data and their semantics using standard formats.

The definition of the source artifact is outlined as follows:

$$
\begin{aligned}
\mathit{src} &= \big(\mathit{id}!, \mathit{d}^*, \mathit{srcType}, \mathit{srcRole}, \mathit{srcIdType}, \mathit{srcIdSpec}, \mathit{format}, \mathit{tool}, @\mathit{rq}^*, @\mathit{r}^*, @\mathit{m}^*, @\mathit{a}^*, \\
&\qquad @\mathit{qm}^*, @\mathit{ent}^*, @\mathit{att}^*, @\mathit{act}^*, @\mathit{param}^*\big) \\
\mathit{srcType} &= \text{'experimental data'} \mid \text{'simulation data'} \mid \text{'literature'} \mid \text{'notebook'} \mid \ldots \mid \text{'other'} \\
\mathit{srcRole} &= \text{'input'} \mid \text{'validation'} \mid \text{'calibration'} \mid \text{'theory'} \mid \text{'illustration'} \mid \ldots \mid \text{'other'} \\
\mathit{srcIdType} &= \text{'DOI'} \mid \text{'URL'} \mid \text{'ISBN'} \mid \ldots \mid \text{'other'} \\
\mathit{srcIdSpec} &= \mathit{text}
\end{aligned}
$$

Case Study: When developing the model, various resources are collected. Among those are research papers that investigated the behavior of lipid rafts in the wet lab. One of them is the publication by Sakane et al., on which requirement Req1 was based on [353]. To make explicit that this publication was used and that the requirement described by it will be used for validation in the simulation study, an artifact can be specified in the conceptual model as follows:

(id = Sakane et al. (2010),
d = [(dType = text,
spec = LRP6 is internalized by Dkk1 to suppress its phosphorylation in the lipid raft and is recycled for reuse)],
srcType = literature,
srcRole = validation,
srcIdType = DOI,
srcIdSpec = https://doi.org/10.1242/jcs.058008,
format = –,
tool = –,
rq = [],
r = [*@Req1*],
m = [],
a = [],
qm = [],
ent = [*@LRP6*, *@Dkk1*, *@CK1γ*],
att = [],
act = [],
param = [])

The simulation study also relies on two existing simulation models [32, 41], thus, two additional source artifacts can be created for the conceptual model for Lee et al. (2003) and Mazemondet et al. (2012). Both are of type literature and their role can be specified as "model used for extension or composition". For brevity these are not shown here but can be specified analogously to the above.

Also, a variety of data have to be collected to pursue the initially defined research question in-vitro and in-silico (see Figure 4.2) and to calibrate and validate the defined model structure. The data collection process is guided by the information already specified in the conceptual model (particularly the qualitative model). First, fluorescence microscopy data of in-vitro neural progenitor cells, e.g., confirms the existence of lipid rafts, and a partial localization of the receptors within these membrane structures (Figure 4.2A). A second prerequisite and important data set for model calibration is obtained by analyzing the protein concentration of β-catenin through immunohistochemistry (Figure 4.2C). These in-vitro data confirm that disruption of membrane lipid rafts in neural progenitor cells attenuates canonical Wnt signaling. Apart from conducting these two experiments in the lab, a variety of literature is consulted, and various conceptual material is collected, such as parameter values from Mazemondet et al. [41] (Figure 4.2B), and the distribution of LRP6 and CK1γ from Sakane et al. [353] (Figure 4.2D). In the conceptual model, for each of the data sets a new source artifact is created, e.g., the information about the β-catenin data (Figure 4.2C) looks like this:

(id	=	Beta Catenin Data,
d	=	[(dType = text,
		spec = Analysis of the protein concentration of β-catenin through immunohistochemistry)],
srcType	=	experimental data,
srcRole	=	calibration,
srdIdType	=	DOI,
srcIdSpec	=	https://doi.org/10.1371/journal.pcbi.1004106.s010,
format	=	CSV,
tool	=	–,
rq	=	[],
r	=	[],
m	=	[],
a	=	[],
qm	=	[],
ent	=	[*@Beta-cat*],
att	=	[],
act	=	[],
param	=	[])

For generating the statistical model checking experiment, these source artifacts are not of immediate relevance. However, there may be other behavioral requirements, given by data. In such cases, the relevant trajectories from the data source artifacts must be incorporated into the experiment specification for comparison with the simulated data.

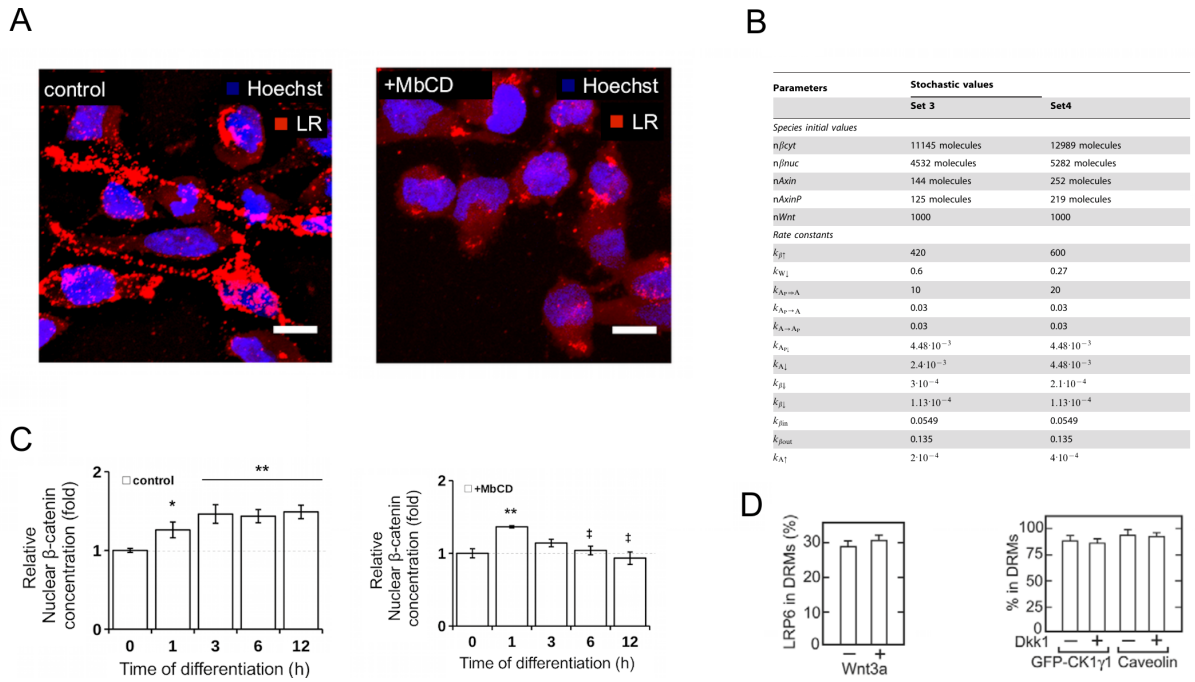


Figure 4.2: Fluorescence microscopy data of lipid rafts (A) collected by Haack et al. [31], parameter table of the intra-cellular Wnt model (B) adopted from Mazemondet et al. [41], relative protein concentration of β-catenin (C) experimentally determined by Haack et al. [31], and distribution of LRP6 and CK1γ in membrane lipid rafts (D) as depicted in Figures 1A and 2C of Sakane et al. [353]. Reprinted from Wilsdorf et al. [83]. © 2020 IEEE.

4.2.4 Implementation

The proposed definition of the conceptual model is exemplarily realized in a prototypical implementation based on *MediaWiki*[39]. Wikis are a popular means for collecting, structuring, and sharing knowledge of all kinds. *MediaWiki* is a free and open-source wiki engine, which allows for simple web-based editing of information with a well-known markup language. Extensions provide additional features, e.g., *Semantic MediaWiki* enables semantic annotations of the knowledge graph [354], and *Page Forms*[40] allows for easy page creation and data insertion via forms.

First, after setting up a wiki named *Conceptual Modeling Wiki (CMoWiki)*, the data structure has to be established. Therefore, categories are created for each artifact type and other substructures such as the conceptual model as a whole (*cm*), the research questions (*rq*), requirements (*r*), descriptions (*d*), etc. Next, the properties for each individual input are created, e.g., format and tool. In addition, the input types need to be defined, e.g., *Text*, *Image*, or *Page* for links to other artifacts. Thereafter, templates are created, which relate inputs to the categories, and define the page layout. Based on the templates, forms can be created, allowing users to edit pages in a graphical dialogue. After these preparatory steps, users can start creating pages and adding data with the *CMoWiki*. For example, the user can add a new page of the category *Conceptual Model* using the corresponding form.

Figure 4.3 shows the main page of the *Wnt Simulation Study*, which contains individual sections for the different artifacts of the conceptual model. In the first section, a research question called *Main Objective* has been created, and further specified on the corresponding page, i.e., by adding a description. Also, a link to a requirement has been added to refine the objective. Since in this snapshot the requirement page of *Req1* has not been specified yet, the link is presented in red. By clicking on the red link, the modeler is directed to the *Create Requirement: Req1* form to enter information about the requirement.

The wiki, including the Wnt case study, is publicly available for illustration and allows the interested reader to export its pages to set up their own wiki[41].

The idea of using a wiki for implementing the conceptual model has various advantages. First, it provides a "GUI" and "language" that are familiar to most scientists and non-scientists. Second, with choosing a wiki, the compromise between formal and informal specification is emphasized, as it grants users the flexibility of using verbal narratives, mathematical formulae, other media, or a combination thereof, as they see fit. Third, it allows for collaborative development of the conceptual model. This is crucial, as the information collected and used during a simulation study is manifold and may need to be clarified and exchanged among modeling experts, data analysts, domain experts, and stakeholders. Last but not least, although the wiki can function as a stand-alone application, it also offers a REST API. This API enables modeling and simulation frameworks to seamlessly write information into the wiki and retrieve information from it as needed, such as during the creation of a simulation experiment. Furthermore, the data model presented here may be easily integrated into other standards or applications, such as provenance editors.

Using the wiki implementation, a conceptual model was built for the simulation study of Wnt signaling by Haack et al. (2015) [31], as discussed in Sections 4.2.2 and 4.2.3. The presented approach to conceptual modeling has also been applied in the area of self-adaptive software systems [355]. The conceptual model definition was integrated into the life cycle of self adaptive systems (SAS), as a means for determining the components of the systems, and making simulation requirements and assurance claims explicit—all of which are crucial in assessing the reliability of the SAS.

[39] `https://www.mediawiki.org`, last accessed 19 July, 2024.
[40] `https://www.mediawiki.org/wiki/Extension:Page_Forms/en`, last accessed 19 July, 2024.
[41] Conceptual Modeling Wiki (CMoWiki), `https://cmowiki.informatik.uni-rostock.de`, last accessed 19 July, 2024.

Figure 4.3: The Conceptual Modeling Wiki (CMoWiki) for creating, editing, and linking artifacts. Reprinted from Wilsdorf et al. [83]. © 2020 IEEE.

4.2.5 Discussion

In this section, the various definitions of the conceptual model, found in the modeling and simulation literature, were integrated. The presented integrated definition provides an approach for making the diverse early-stage products of a simulation study (i.e., research questions, requirements, qualitative models, assumptions, methodologies, data and information sources) explicit. Relations between individual artifacts can be made explicit as well by creating references. The approach advocates a broad interpretation of the conceptual model in modeling and simulation. It supports managing the documentation of context and automation during the simulation study in a structured manner, and facilitates later assessment and reuse of a simulation study and its results.

Each early-stage product is captured by 1) some form of multimedia description, 2) the "product" itself, using formal languages for specification where applicable, and 3) further means that help interpreting the product. Thus, in the ideal case, products ship not only with a formal syntax but with a semantics that enables an automatic interpretation by respective tools for acting upon this semantics.

Furthermore, if detailed information is made explicit, structured, (partly) formalized, and assigned with semantics the artifacts of the conceptual model can be computationally exploited to support modelers in their simulation study. In particular, the possibilities for automatic experiment generation were discussed. In the case study, the conceptual model with formally specified requirements was pivotal for automatically generating and executing a statistical model checking experiment. Another example would be the unambiguous documentation of model parameters, which may be used for automatically setting up a sensitivity analysis [220]. Moreover, the conceptual model can be crucial in automatically deciding which type of experiment to run, and also to adapt a previously executed simulation experiment for a given situation, which will be described in detail in Chapter 5.

Recording, maintaining and exploiting the different aspects of the conceptual model promises to be particularly useful if various simulation models shall be developed. It reveals implicit relations, as well as similarities and differences between the early-stage products of the individual studies. Therefore, sharing conceptual models, e.g., through a wiki, can improve collaborative development and community reuse of artifacts.

However, the question remains whether conceptual modeling itself can also be automated.

Recently, systems for automatic code documentation have significantly advanced and allow generating documentations including code examples, tutorials, and block and inline comments by utilizing various techniques for natural language summary generation [356] as well as transformer-based large language models (LLMs) [357]. In M&S, LLMs have been used to generate executable simulation models from natural language descriptions [358]. In the near future, similar approaches could be investigated for conceptual models so that wikis, diagrams of qualitative models, or reporting documents would be generated instead of manually created. However, it is unclear what the precise input of such a procedure would be, and whether not conceptual modeling will remain the one key part of the M&S life cycle that needs to be driven by humans.

Most importantly, conceptual modeling is essential for defining the domain and scope of the problem, specifying project requirements, providing context, and planning the modeling and analysis steps ahead. It thus is an essential part of the iterative modeling process (as defined in Section 2.1), as it guides or even prescribes the actions to come but also is refined by those actions. Furthermore, the conceptual model can be seen as a "thinking tool" for developing understanding of a domain or a particular problem. To ensure that the conceptual model can fulfill its role as a core artifact of a M&S study, additional user support for conceptual modeling is essential.

The current implementation of the broad and formalized definition in a wiki provides flexibility to modelers, offering guidance based on a defined structure for the different types of artifacts. However, to maximize the value of explicit conceptual modeling for users, additional support may be required, including an integration with existing modeling and simulation software. This integration is also crucial for collecting practical experiences and further case studies that help to refine the definition of the conceptual model, probe additional methods for formalization, and evaluate their implications for conducting a simulation study more systematically and effectively. In addition, to motivate a more widespread adoption of explicit conceptual modeling and the development of appropriate tools, the broad and (semi-)formal approach for conceptual modeling needs to be integrated with other approaches for supporting simulation studies. For instance, it can serve as a data model for tracing provenance during the simulation study [28], and as input for automatic experiment generation.

4.3 Making Relations between Products, Activities and Studies Explicit

Developing a valid simulation model to explain, analyze or predict real-world processes (e.g., of a cell biological system) is an intricate task. This task is growing in complexity as it involves the extension and composition of models, utilization of various input data, and the execution of various simulation experiments needed to calibrate, validate and analyze the model.

So far, this chapter has examined how to make various artifacts of a simulation study explicit, with particular focus on the conceptual model. However, this alone does not suffice for enabling a comprehensive understanding of the entire story behind a simulation study, and for supporting the reusability of artifacts for generating simulation experiments. Equally crucial is making the relations of the main products explicit, and explicitly documenting how the different artifacts were produced or used by activities during a study. During experiment generation, e.g., the following questions about the relations in a simulation study need to be answered automatically:

- How was the model developed and based on what assumptions?
- What analyses were conducted with the different model versions?
- What data was involved in model calibration or validation?
- How and when was the model curated?

- Who and what tools were involved in which steps?
- How is the model related to other simulation studies?

Consequently, approaches are in demand for documenting the entire simulation study from beginning to end, and all the activities that contributed, while also ensuring that the information is machine-accessible and queryable.

The wiki presented earlier in the context of conceptual modeling already represents a knowledge graph for relating artifacts to each other. E.g., it allows expressing what parts of the qualitative model are mentioned in an assumption, or what approach a simulation model refers to. However, to comprehensively trace the modeling and analysis steps in which these artifacts were involved or by which they were produced, a different perspective is required.

Documentations via reporting guidelines are not suited to meet this challenge, as they typically focus on the "final" product (i.e., a simulation model) of the simulation study rather than describing the steps taken to reach that endpoint. Workflow systems, on the other hand, can support the process of creating the various artifacts by planning the steps ahead according to some life cycle definition. However, without additional mechanisms for provenance capture, their "prospective" approach is inadequate for capturing the actual occurrences during the simulation study and the precise relationships between the products and activities.

In contrast, retrospective provenance via provenance graphs has been introduced to formalize the relations between different kinds of products and activities explicitly. Provenance, in general, refers to the "information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability, or trustworthiness" [359]. The provenance data model (PROV-DM), in particular, provides a formal, graph-based representation, which facilitates the visualization and formal analysis of how products evolved.

However, various challenges are associated with provenance, such as how to integrate it with other forms of model documentation, how to integrate provenance with existing tools for conducting and managing modeling and simulation studies, and how to acquire the provenance information.

A recent effort towards combining provenance graphs with another form of model and study documentation is ODD+P [360]. There, protocols based on the reporting guideline ODD (Overview, Design concepts and Details) for agent-based models were enriched with provenance information following the Open Provenance Model (OPM). In terms of integrating and sharing provenance with existing research infrastructure, provenance can be integrated as part of research articles or their supplementary materials. In terms of integration with modeling and simulation software, provenance is often paired with workflow systems, where provenance traces are automatically recorded as users move through the stages of a workflow [361, 299]. Other approaches have looked at extracting provenance without an explicit workflow, emphasizing the growing demand for provenance documented in a standard like PROV-DM. These techniques include the automatic capturing of provenance graphs from scripts based on the structure of the code and execution logs [362], user annotations [363], or analyzing the structure of electronic lab notebooks [364]. First approaches also used manual extraction of PROV-DM graphs from scientific publications to reveal the relations between a family of simulation models [43].

In practice, the artifacts of a simulation study often live in different (online) repositories, such as GitHub, or are published in a model database. Therefore, repositories and model databases need to be considered when looking at provenance as they already contain an ample collection of source files and reports, accompanied by metadata. However, the way information is currently presented has certain limitations regarding what information can be exploited automatically, and how the design of repositories and databases can be tailored to the demands of an automatic experiment generation. The integration of PROV-DM with model databases therefore would be a substantial addition to the state of the art, as it would allow for the story of a simulation study

to be documented according to the FAIR (findable, accessible, interoperable, reusable) principles, and for the provenance graphs to be computationally assessed. Moreover, model databases are widely used in their respective modeling communities, and thus integration with provenance graphs could boost reproducibility and reusability for a plethora of existing and future models. BioModels [42], for instance, is currently the largest database of models in systems biology, and the largest database of curated computational models overall.

This section first discusses the state of the art regarding approaches that allow making the relations between artifacts, activities and studies explicit. It then introduces a case study involving the BioModels database and a family of cell signaling models. In the main part, using illustrating examples from the case study, it is investigated how the information contained in a model database like BioModels can be represented using PROV-DM to make the relations accessible and how the simulation model(s) was(/were) developed and analyzed. The central question is, how the model database can conform to PROV-DM and what steps and modifications are necessary to create these formal provenance representations automatically. Finally, the generated provenance graphs are explored based on graph queries to gain additional insights into the simulation studies and to automatically generate simulation experiments. The case study demonstrates the value of provenance representations in enhancing the automatic interpretation of studies while also elucidating the connections between studies. Especially the involved sources can be more easily identified and consistency checks can be carried out. Furthermore, the relations between different models as well as the model iterations and curation history will be illuminated by provenance.

Implementing the suggested automatic procedure for creating provenance graphs as additional views within different model databases would immediately expand these benefits to numerous modeling communities. This would enable them to establish connections between their models more easily, even when dealing with "legacy" simulation studies, a term referring to simulation studies that have already been published without explicit provenance information.

This section has been adapted with modifications from the publication [302]. It was extended by a discussion of the state of the art in making the relations of artifacts, activities and studies explicit and accessible. In addition, a case study of cell signaling models was moved into focus, queries were introduced to illustrate how the provenance graphs can be explored both interactively and automatically, and further discussions were added. Furthermore, a software prototype is provided.

4.3.1 State of the Art

Various approaches exist that allow making the relations of artifacts, activities and studies explicit and accessible. These include reporting guidelines, workflows, provenance, and model repositories and archives.

Reporting Guidelines

In the previous section (particularly Section 4.2.1), a variety of reporting guidelines were discussed with respect to the (early stage) products and metadata of the simulation study they consider. Most of them focus on the documentation of a single ("final") simulation model and the context in which it was built, and therefore do not request making the relations between products, activities, and multiple model iterations and studies explicit. The purpose of ODD, for instance, is specifically not to show the evolution and relation of products. Grimm et al. state that "[t]here certainly is also the need to describe a model's underlying story, or narrative, but ODD is not the place for this" [75].

Examples of documentation guidelines that partly aim at documenting all the essential steps, sources, and products of the modeling and simulation life cycle, are TRACE [236] and STRESS [246].

Table 4.2: Structure and contents of TRACE documents (elements 5–8), verbatim from Grimm et al. [236]. For elements 1–4, refer to Table 4.1.

TRACE element	This TRACE element provides supporting information on:
5. Implementation verification	(1) Whether the computer code implementing the model has been thoroughly tested for programming errors, (2) whether the implemented model performs as indicated by the model description, and (3) how the software has been designed and documented to provide necessary usability tools (interfaces, automation of experiments, etc.) and to facilitate future installation, modification, and maintenance.
6. Model output verification	(1) How well model output matches observations and (2) how much calibration and effects of environmental drivers were involved in obtaining good fits of model output and data.
7. Model analysis	(1) How sensitive model output is to changes in model parameters (sensitivity analysis), and (2) how well the emergence of model output has been understood.
8. Model output corroboration	How model predictions compare to independent data and patterns that were not used, and preferably not even known, while the model was developed, parameterized, and verified. By documenting model output corroboration, model users learn about evidence which, in addition to model output verification, indicates that the model is structurally realistic so that its predictions can be trusted to some degree.

TRACE structures the documentation according to crucial activities in conducting a simulation study, including problem formulation, model description, data evaluation, conceptual model evaluation, implementation verification, model output verification, model analysis, and model corroboration. It thus focuses not only on the simulation model as the main artifact, but also on what the conceptual model entails. It also goes beyond early stage products (which were presented earlier in Table 4.1) by documenting how those are used in simulation experiments and, in particular, how the various products were obtained and evaluated. It contains elements for describing simulation experiments and their contexts, see Table 4.2: Element 6 refers to model calibration, whereas element 7 refers to model analysis and element 8 to model validation. Also element 5 (model verification) may involve simulation experiments.

STRESS also includes descriptions of the modeling and simulation process. Central points on their checklist are how data was prepared, how simulation experiments were conducted and what the aim of experimentation was, and how the model was actually implemented.

Another example is the documentation framework for agent-based models by Triebig and Klügl [365]. The format they propose aims to guide the structuring, efficient search, and navigation of the documentation. It consists of documentation blocks referring to the model development cycle. The blocks comprise A) model metadata, B) model description, C) model content, D) expectations on model output, E) experimental frame, and F) passed tests. These blocks serve different purposes within the documentation, i.e., description of the analysis of objects, model concept development, model implementation, model calibration and testing, and deployment runs. For instance, blocks A), D), E), and F) together fully describe a model calibration. This emphasizes the aim of the reporting guideline to also describe the relations of products and steps depending on their role in the modeling and simulation life cycle. An XML-based structure is proposed to capture the different blocks and their metadata.

Regarding the reporting of simulation experiments, MIASE specifies Minimum Information About a Simulation Experiment [247]. This includes the precise settings applied when executing the model, including all parameters, algorithms, and post-processing of the output. In addition, MIASE allows experiments to be explicitly linked to the respective model specification they execute, and to reference ontologies that clarify which settings were used. It thus focuses on

the role and relations of a particular simulation experiment within the simulation study. The external domain-specific language SED-ML implements the MIASE reporting guidelines, thereby helping modelers in adhering to these guidelines [193].

Further guidelines that focus on reporting simulation experiments and their context are MSRR and PSRR [245]. The former comprises the experiment specification as well as pre- and post-processing of data, whereas the latter adds information about sensitivity analysis and model uncertainty. This is augmented by the Minimum Optimization Reporting Requirements (MORR) and Preferred Optimization Reporting Requirements (PORR), guidelines specifically designed for the documentation of optimization experiments. While those reporting guidelines do not explicitly provide means for linking and telling the whole story of a simulation study, they may present important building blocks of an all-encompassing documentation, which may involve the use of various reporting standards in combination.

Other reporting guidelines also focus on specific aspects of a simulation study, such as the collection of empirical data with STROBE (Strengthening the Reporting of Observational Studies in Epidemiology [366]) or the usage of data with RAT-RS [324]. The RAT-RS guideline explicitly provides a set of questions that encourages describing the narrative of the entire simulation study. For instance, the modeler is asked what previous models were used and how, what data was used during the modeling process and for what purpose, what types of experiments were run, and how the output data did support the research question?

Overall, different reporting guidelines cover different aspects of a simulation study. Some focus solely on the documentation of artifacts, while others also ask for detailed descriptions of how these where developed or used. These detailed descriptions may even discuss the interdependencies of the various products, activities, and other studies. However, the reports often exhibit the typical shortcomings associated with reporting guidelines. They require substantial effort from the users and result in lengthy documents. Moreover, these reports predominantly consist of verbal narratives, assisting human readers in deepening their understanding of a study but preventing easy automatic exploitation of the contained information. Therefore, combining reporting guidelines with structured, and ideally formalized, documentation approaches would prove valuable. A promising approach in this direction is ODD+P, which enriches ODD documents by means for formally specifying the relations between artifacts and activities in a standardized provenance model. It thus explicitly caters to the question of "how" simulation models and their related artifacts were developed [360], and makes reporting guidelines (at least partly) exploitable for automatic experiment generation.

Workflows

Workflows present reusable blueprints for sequences of tasks that can be carried out by a user or system. Workflows can provide various benefits, e.g., they allow easy and frequent execution of the same or a similar sequence of steps, which may also be shared with and re-run by collaborators or reviewers [367]. In addition, workflows play a crucial role in expanding the scalability of tasks in distributed environments and in facilitating their semi-automated execution. Valuable applications of workflows can be found in the business domain and the scientific domain [368].

Business workflows are a long-standing research field, with a history extending over two decades. These workflows are defined as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [368]. These tasks may refer to the complete sequence of steps required to process a purchase order or internal enterprise tasks like bookkeeping. Standards for business process management include the Business Process Modeling Notation (BPMN) [369] and the Business Process Execution Language[42] (BPEL).

[42]Web Services Business Process Execution Language (WSBPEL) version 2.0, 2010. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`, last accessed 19 July, 2024.

These concepts from business process management have also been transferred to the scientific domain. Scientific workflows have been established to streamline a wide range of data processing tasks and intricate calculations [367]. Notable examples of existing workflow management systems designed for defining and executing these data-centric scientific workflows include Kepler [370] and Taverna [371].

Scientific workflows as well as business workflows are often realized as imperative workflows. In imperative workflows, all possible paths through the workflow are explicitly specified by activities and control structures. Thus, imperative workflows are most suitable for processes where only few deviations from the default execution exist [372]. In simulation studies, those activities may refer to distributed computing, executing the steps of an individual simulation experiment, running multiple replications of a simulation, and the processing of input or output data [223]. Relations between artifacts and activities are therein specified implicitly by the order of the imperative statements and the data the workflow uses and produces. The procedural rules, however, are usually tailored to the particular project and data, and do not allow for flexibility in the processes. In business workflows, thus, a second type of workflow has been introduced.

Declarative workflows, such as artifact-based workflows based on the Guard-Stage-Milestone model [373], describe possible activities that can be taken by a user (known as stages), their preconditions (known as guards), and postconditions (known as milestones) to be achieved, and are a common approach in modeling business workflows [374]. In contrast to their imperative counterparts, declarative workflows usually follow an open world assumption, meaning that everything is allowed unless it is explicitly prohibited by the guards [372]. This provides additional flexibility in the processes to be supported, as it only is meant to prevent the users of the workflow system to do something wrong but not restrict them generally in what they do or how.

Like many business processes, simulation studies consist of such loosely coupled tasks where modelers are going back and forth between the steps of the M&S life cycle. Thus, declarative workflows allow supporting entire simulation studies [223]. By their preconditions and postconditions, these workflows can relate the artifacts and activities of a simulation study to each other. Additionally, the artifact-based paradigm allows references to related artifacts to be explicitly stored in the information models of the respective workflow artifacts. Previously, Ruscheinski et al. used these conditions (guards and milestones) and relations to re-run simulation experiments for re-validation or re-calibration [223].

The declarative workflow design also facilitates the extension to domain-specific specializations of the workflow, e.g., for finite element simulation studies [52]. The domain-specific information in the information models can aid in generating simulation experiments, e.g., if specific information about error bounds is given [52]. An example of this was shown in Section 3.5.4.

Workflows generally (business or scientific, imperative or declarative) can be considered "prospective provenance". This type of provenance prescribes what should or should not be done during a simulation study and plans the steps ahead [375]. In contrast, retrospective provenance is concerned with how a simulation study was conducted. It requires precisely recording the steps taken and artifacts used or produced during the simulation study (with or without workflow execution). Retrospective provenance therefore is suited for making the story of an entire simulation study explicit, which will be discussed next.

Provenance

Retrospective provenance (in the following just referred to as "provenance") is described by the W3C Provenance Working Group as the provision of "information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability, or trustworthiness" [359]. Referring to simulation studies, the term "provenance" subsumes, among other information, process details about the construction of the conceptual model, the development, extension, and composition of submodels, as well as the

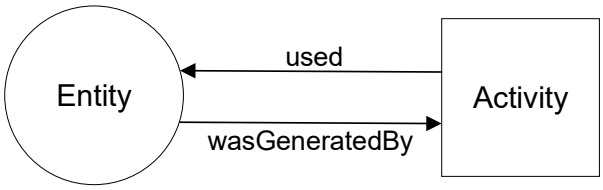


Figure 4.4: PROV-DM graphical notation, adapted from Belhajjame et al. [377].

calibration and validation experiments conducted. Provenance therefore is crucial in making the context and the entire story of a simulation study explicit, revealing how the different artifacts evolved over time.

Similarly to the Open Provenance Model (OPM) [376], the provenance data model (PROV-DM) of the PROV family of standards allows representing provenance information in terms of the types *entitiy* and *activity* [377]. Entities and activities can be connected via relations, with *used* and *wasGeneratedBy* being the most common relations. Figure 4.4 depicts these core provenance types and the relations between them. In addition to those, entities and activities can be associated with agents to indicate, e.g., responsibilities of different scientists during the project or "software agents".

The general nature of these provenance types allows for PROV-DM to be applied to and specialized for a wide range of application fields in computer science and beyond, e.g., to model information diffusion in social media [378] or to analyze traffic on the internet of things [379].

With respect to simulation studies, PROV-DM has been customized using the entity types simulation model, simulation experiment, wet-lab data, and simulation data [28, 380]. This specialization was further refined with simulation models of signaling pathways in mind [43] and based on discussions about the role of conceptual modeling in simulation studies [83]. In this refined "provenance ontology", the entity types encompass early-stage products of the conceptual model, such as research questions (RQ), assumptions (A), requirements (R), and qualitative models (QM), in addition to simulation models (SM), simulation experiments (SE), wet-lab data (WD), and simulation data (SD). Activities are also refined and refer to building (BSM), calibrating (CSM), validating (VSM), and analyzing (ASM) the simulation model. Further extensions exist for social [301] and ecological [381] simulation studies. In the former, for instance, participant information, preregistration, and ethical approval are included as crucial entities during primary data collection, whereas for the ecological modeling and simulation studies, field data and visualizations have been included as well as some domain-specific metadata.

In all the aforementioned data models, agents are neglected since modeling projects are often conducted by a single individual. However, in the case of large collaborative studies, where different parts of the study may be conducted by different researchers or even teams of researchers, including agents in the provenance graphs can prove useful. In the following sections, agents are also not explicitly represented in the provenance graphs as this information can be easily integrated in the metadata of the entities and activities. However, if further agent roles (e.g., laboratory technician, data analyst, simulation specialist, or visualization expert) are added in the future, this decision can be reconsidered. Similarly, if software agents play an increasing role (e.g., if automatic experiment generators act as autonomous agents in the simulation study), these may be included as well.

PROV-DM provides an intuitive graphical representation, where entities and activities are represented by oval and rectangular shapes, respectively, and the relationships are represented by directed edges, see Figure 4.4. This facilitates exploring the simulation studies manually, and visualizing them via web tools. Figure 4.5 (top layer) shows an exemplary provenance graph of a small simulation study. Reading the graph from left to right with the arrows showing back in time, it first contains a "building simulation model" (BSM) activity that was based on a research question, an assumption, and a qualitative model description. This is followed by a "calibrating

simulation model" (CSM) activity that uses a requirement to fit the model and produces a simulation experiment, simulation data, and a calibrated model.

Under the hood, provenance recorded according to PROV-DM forms a directed acyclic graph, where entities and activties are nodes and relationships represent the directed edges. This formal representation enables provenance to be analyzed automatically by graph algorithms and graph queries. For instance, if the provenance graph is stored in a graph database, such as Neo4j [382], a query language (e.g., Neo4j's Cypher [383]) can be used to extract interesting subgraphs, providing insights into the process and possibly detecting inconsistencies. Using Cypher, for instance, specific paths, individual nodes, or metadata of nodes can be queried and filtered by properties, types, or patterns. Matched results can be bound to variables and compared or returned by the query.

Various insights can be gained by navigating and querying provenance graphs. Overall, Herschel distinguishes seven purposes of provenance [384]. In the context of simulation studies, those can be summarized as follows: *Collaboration*: In collaborative simulation studies, provenance enables modelers to synchronize their work effectively. *Presentation*: It enhances the presentation of information, making it easier to visualize relationships between products, activities, and, when necessary, agents in a simulation study, thus improving overall understandability. *Attribution*: Provenance clarifies authorship and ownership of the various parts within a study, helping to identify individuals responsible for products and their content. *Recall*: Provenance serves as a kind of logbook, allowing modelers to recall their steps and track the evolution of their work. *Replication*: Independent researchers can utilize provenance for replicating study results, improving traceability and credibility of the research. *Process quality*: Provenance plays a crucial role in assessing the quality of a simulation study, referring to the correctness of the performed activities. Provenance can also assist in the design of future simulation studies and thereby improve their quality. *Data quality*: Provenance also allows assessing the quality of the data used or produced by the simulation study, e.g., w.r.t. completeness, accuracy, and trustworthiness of the source.

To serve these diverse purposes, metadata stored inside the provenance nodes is just as crucial as the information encoded in the graph structure itself. Metadata is indispensable for interpreting what a product and its contents are about. For instance, in the case of a simulation model, metadata includes a reference to the model specification. Similarly, a simulation experiment entity includes a reference to the file(s) containing the experiment specification, in addition to details about the used experiment type. The metadata of conceptual entities, on the other hand, can be based on definitions of the conceptual model (as discussed in Section 4.2, and prior works [43, 83]). Assumptions, for example, can be characterized by the language in which they are defined, along with associated domain-specific ontology terms.

The practical value of provenance comes from the tools designed around PROV-DM. Various web editors and graph databases are available for editing, storing and exporting provenance data [385]. Additionally, the integration of provenance with version control systems like Git has been discussed, especially concerning the management of source code across different versions [28]. Moreover, the usefulness of provenance is determined by its level of detail. Different granularities have been explored as well as aggregation- and sequence-based abstractions [386, 299]. The most fine-grained level of provenance provides information about each minor modification (e.g., each time the model specification was saved), whereas the top level provides a bird's eye view of the major milestones in model building, validation and calibration. These aggregation levels are illustrated in Figure 4.5. The bottom layer shows sequences of changes recorded in the working environment of the user. These may be as small as simply tweaking one model parameter, renaming a model entity, or selecting the type of requirement. Higher-level, aggregated provenance views can be derived automatically from this [299]. At the highest level, on the other hand, a bird's eye view of the study is provided, showing "only" the major steps in terms of when a model was refined, calibrated, validated, or analyzed. The choice of granularity depends on the

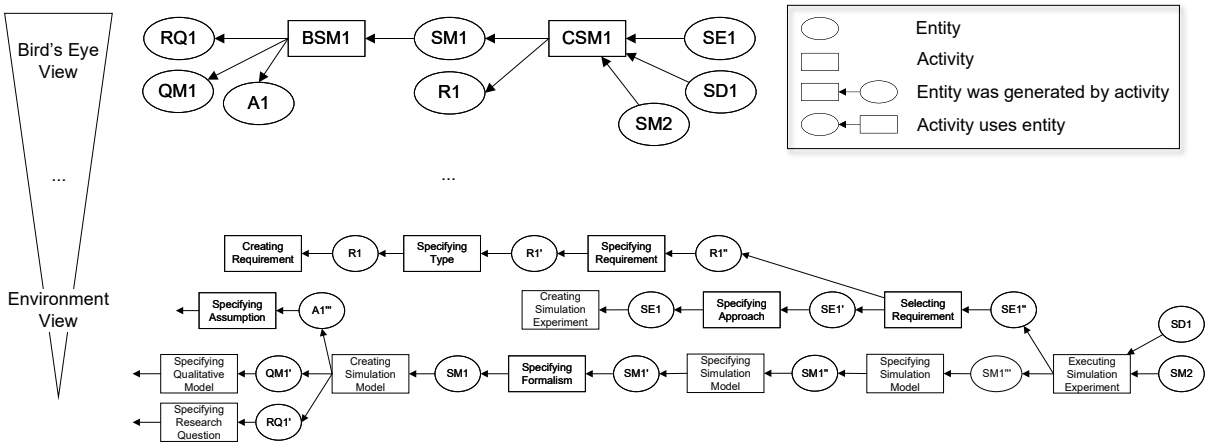


Figure 4.5: Various views on an exemplary provenance graph of a simulation study in the PROV-DM graphical notation. The bottom layer shows fine-grained provenance as, e.g., recorded by an artifact-based workflow system or by observing a modeler in their working environment. It shows sequences of creating, specifying, and interlinking the various entities of a simulation study (including RQ – Research Question, QM – Qualitative Model, A – Assumption, R – Requirement, SM – Simulation Model, SE – Simulation Experiment, SD – Simulation Data). Thereby, various intermediate versions of the entities are produced. The top layer shows a bird's eye view depicting aggregated activities related to building the simulation model (BSM) and calibrating the simulation model (CSM) and their respective inputs and outputs.

intended use of the collected provenance information. In this dissertation, a rather coarse-grained provenance model will be the foundation for automatically generating simulation experiments. This level of detail is sufficient for conveying the story of a simulation study, and for reasoning about subsequent or alternative model versions as well as the calibration, validation, and analysis experiments conducted. Furthermore, the used provenance model is suitable for elucidating the context in which those various activities were conducted, and also the relations between a family of simulation models [43]. All of this information is crucial for automatically making decisions about when to conduct which simulation experiments and how.

Numerous methods exist for capturing provenance, with many of them originating in the context of scientific workflows [387]. In imperative scientific workflow systems, such as Kepler and Taverna, provenance information is captured implicitly via event logs [361, 388]. Another essential technique involves code annotation, where tools like YesWorkflow enable modelers to annotate specific statements in their scripts to be tracked [363]. Other methods operate on the level of the working environment or operating system without requiring annotations from the user. For instance, NoWorkflow is a provenance capturer that employs techniques such as abstract syntax tree analysis, reflection, and profiling, and therefore does not require users to modify their scripts [362]. Similarly, Starflow uses code introspection to inspect scripts for specific keywords, employing both static code analysis of the control flow, and dynamic analysis by tracing function call stacks and I/O operations [389]. In the context of declarative, artifact-based workflows, provenance can be captured through patterns [299]. Recent advances allow capturing provenance transparently and non-intrusively by observing the modelers in their familiar environment, and by observing function calls and system calls [300]. However, it is important to note that provenance comprises information of conceptual nature, such as assumptions and requirements. These aspects of the conceptual model are notoriously difficult to track automatically if users do not document them explicitly [300]. Therefore, for comprehensive provenance recording of the entire modeling process, there will always be a manual component. Nevertheless, this can be supported by dedicated software with an editor capable of requesting these conceptual artifacts and their

associated metadata on-demand through an easy-to-use (web) interface [300].

Model Repositories and Archives

Automatically generating simulation experiments requires the artifacts of a simulation study to be available and machine-accessible. First, they are needed as context to inform and steer the experiment generation. Second, existing experiment specifications need to be reused and adapted to the given context.

Recently, the FAIR (findable, accessible, interoperable, reusable) principles for open research [250, 390] have gained increasing attention and have been adopted across various scientific communities. Their commitment to FAIR research also aligns with good modeling practices, and the goals of reproducibility and reusability—all of which are increasingly recognized and valued by academic journals, including PLOS Computational Biology[43] and the ACM Transactions on Modeling and Computer Simulation[44].

Model repositories and archives can contribute to implementing the FAIR principles for simulation studies. In particular, they allow making artifacts findable (F) and accessible (A), which is crucial for automatically exploiting information. But they may also assist in improving the interoperability (I) and reusability (R) of studies, depending on what kind of information is represented in these databases and how.

So far, online repositories for modeling and simulation, like many model documentation formats and guidelines, focus primarily on the simulation model. Those comprise the BioModels [42] database, which specializes in mathematical models in systems biology and currently contains over 2500 models, and the CoMSES[45] model library, formerly known as OpenABM, which is a repository originally designed for agent-based models and hosts a collection of over 1000 published models. Examples also include the NetLogo Model Library[46], which is maintained by the NetLogo agent-based modeling community, and a fairly recent attempt aiming to share so-called Reusable Building Blocks[47], i.e., components with a specialized functionality that may be useful within a multitude of models.

Besides publishing the simualtion models and analysis scripts, open scientific libraries have played a pivotal role in sharing a wide range of other scientific resources. Notable examples include the GeneLab Data Systems (a platform dedicated to sharing of vast "omics" data sets, encompassing genomics, transcriptomics, proteomics, and metabolomics) [391], OpenNeuro (a platform specifically designed for sharing neuroscience data, including MRIs [392]), governmental initiatives towards open data (e.g., various COVID-19 datasets compiled by the USA's National Center for Health Statistics[48]), or myExperiment (a platform that serves as a hub for sharing scientific workflows [393]).

Research artifacts may also be made available using general purpose repositories, such as GitHub[49] and ZENODO[50], or via the supplementary material of an open access publication in a scientific journal. The latter two generate digital object identifiers (DOIs) that allow referencing the artifacts persistently.

More specifically tailored to scientific artifacts of a particular domain are certain containers and archives [253]. Through the bundling of artifacts and metadata, they allow partly making the relation of artifacts explicit as well as their role during the simulation study. COMBINE (COmputational Modeling in BIology NEtwork) archives, for instance, are standardized containers

[43] `https://journals.plos.org/ploscompbiol/`, last accessed 19 July, 2024.
[44] `https://dl.acm.org/journal/tomacs`, last accessed 19 July, 2024.
[45] `https://www.comses.net/`, last accessed 19 July, 2024.
[46] `https://ccl.northwestern.edu/netlogo/models/`, last accessed 19 July, 2024.
[47] `https://www.rbb4abm.com/`, last accessed 19 July, 2024.
[48] `https://www.cdc.gov/nchs/covid19/`, last accessed 19 July, 2024.
[49] `https://github.com/`, last accessed 19 July, 2024.
[50] `https://zenodo.org/`, last accessed 19 July, 2024.

for bundling files related to a simulation study in computational biology. They require the original publication, the (final, published) simulation model, the analyses, as well as metadata to be provided using standardized domain-specific formats. To those formats belong SBML for the model specification [283], SED-ML for specifiying the analyses [193], and metadata describing the provided files using the Open Modeling EXchange format (OMEX) [253]. Another approach present the reproducible Workflow Research Objects (WRO) that encapsulate scientific workflows and all associated context information, scripts and other resources [254]. In addition, provenance traces, e.g., produced by YesWorkflow, may be included. Alternatively, computational notebooks like Tellurium can compile different artifacts in an executable manner [394]. As Tellurium notebooks align with the standards required by COMBINE, it is possible to create COMBINE archives from these notebooks.

However, despite sharing collections of artifacts, there is no guarantee that these resources can fully reproduce the study's results, often due to erroneous scripts, missing dependencies, and other factors. For this reason, BioModels and CoMSES offer peer review processes conducted by independent researchers, allowing for a thorough examination of the artifacts' quality and reproducibility. In addition, badges may be awarded for accrediting quality to the published code. The ACM, for instance, offers reproducibility badges[51] to state that the artifacts published with a journal article or conference paper are functional, reusable and available, and that the main results of the paper could be reproduced. The Open Code badge[52], as employed by CoMSES, poses another option for rewarding researchers if their artifacts meet certain criteria. Furthermore, the Open Data[53] badge issued by the OSF (Open Science Framework) may be assigned to the various datasets and materials.

When it comes to publishing entire simulation studies, model databases and archives mainly contribute by making artifacts findable and accessible. Their capabilities (also with respect to experiment generation) are currently limited by the fact that the relationships between artifacts, activities, and studies are only partially made explicit. For example, understanding the connections between different model versions and the analyses performed with them, as well as the order in which they were executed, may not be apparent when individual artifacts are shared. Furthermore, these repositories and archives typically do not make activities explicit, except in cases where scientific workflows are shared. Moreover, while simulation studies may be related to one another through semantic tagging or explicit linking of web pages (in the metadata), there is often a lack of explicit information on how this related information is utilized.

Therefore, to capture entire simulation studies and the information necessary for automatic experiment generation, it is imperative to bridge model databases with other forms of documentation, as they can provide complimentary information. By integrating the various documentations, different views of a simulation study may be offered depending on the needs of the user or software accessing it. For instance, referring to reporting guidelines, organizations like CoMSES already encourage modelers to provide ODD reports alongside their artifacts[54]. Also, for small recurring tasks, such as data processing, the combination of (scientific) worfklows and online repositories has been explored [393]. Provenance graphs, however, have not yet been integrated within model databases as a central form of information representation.

Therefore, an interesting challenge ahead is to seamlessly integrate provenance based on PROV-DM with the models, experiments, data, code and other sources stored in model databases. The following subsections will explore the potential of this combination to automatically gain insights into a simulation study and to generate simulation experiments. The utility of provenance graphs in making the relations between artifacts, activities and related studies explicit and exploitable will be demonstrated via a case study.

[51] `https://www.acm.org/publications/policies/artifact-review-and-badging-current`, last accessed 19 July, 2024.
[52] `https://www.comses.net/resources/open-code-badge/`, last accessed 19 July, 2024.
[53] `https://osf.io/tvyxz/wiki/1.%20View%20the%20Badges/`, last accessed 19 July, 2024.
[54] `https://www.comses.net/resources/guides-to-good-practice/`, last accessed 19 July, 2024.

4.3.2 Case Study

Analyzing and understanding complex biological processes of interacting subsystems requires simulation models to be available for reuse by other researchers. BioModels is a platform that facilitates the sharing of FAIR simulation models [42]. The database is free and openly accessible[55].

Most models in BioModels are ordinary differential equation models, but recently also other model types, e.g., logic-based or constraint-based models, are supported. Models have to be encoded in standardized formats such as SBML [283] or CellML [395]. From the model files, reaction network diagrams can be generated to be also made available. Increasingly, also the simulation experiments are shared in separate formats, e.g., using SED-ML [193] or COPASI [194]. The COMBINE community initiative coordinates the development of the various standards and their combination, e.g., to bundle all information needed to reproduce a simulation experiment, such as data, simulation model, or simulation experiment specification in an archive [253].

Models submitted to BioModels must adhere to the MIRIAM (Minimal Information Requested in the Annotation of Models [320]) reporting guidelines. MIRIAM requires modelers to include several pieces of metadata with at least a unique name, a citation associating the model to a publication, contact information for the model authors, the date and time of model creation and last modification, and the terms of distribution. Beyond that, to unambiguously identify the model components, models can be semantically annotated and linked to ontologies and other databases like the NCBI Taxonomy [396], the Gene Ontology [397], or KEGG [398]. Furthermore, they can be cross-referenced with other models.

The models submitted to BioModels are independently curated to ensure that they are consistent with the referenced publication and that they produce the described numerical results. Over the past years, BioModels has become the world's largest repository of curated computational models. Currently (as of July 2024), the database counts over 2500 published models, of which more than 1000 have been manually curated.

To illustrate the concept of combining model databases with explicit context information about a simulation study in the form of provenance graphs, a family of cell signaling models is used as a running example. This includes at its center a simulation study of the ERK, PI3K/Akt and Wnt signaling network by Padala et al. [399]. The model can be found in the BioModels database under the identifier BIOMD0000000652[56]. It represents a signaling network, i.e., a set of molecular reactions that control cell functions such as cell growth and cell death. Disruptions in this signaling network can cause malfunctions of the cells, leading to cancer growth. In their simulation study, Padala et al. aimed to understand the exact mechanisms that lead to cancerous cell function, and to quantify the impact of various perturbations of this network. In this endeavor, their study is linked to various other simulation studies, forming a family of cell signaling models. Results and experiences from the related simulation studies are of particular interest for simulation experiment generation.

Figure 4.6 shows the BioModels page of this model. It comprises five areas: "Overview" provides a description of the model as well as semantic annotations via domain-specific ontologies. "Files" shows all the files that were uploaded, preferably in standard formats of the systems biology community. "History" allows accessing earlier versions of the database entry, including their files and metadata. "Components" lists the central species and reactions, as well as initial concentrations and parameter values. And "Curation" provides information about what figures of the corresponding research paper could be reproduced by an independent reviewer using the provided source files.

The following first discusses how provenance graphs are constructed from the information given in BioModels, detailing the available provenance entities, activities, and metadata that

[55] `https://www.ebi.ac.uk/biomodels/`, last accessed 19 July, 2024.
[56] `https://www.ebi.ac.uk/biomodels/BIOMD0000000652`, last accessed 19 July, 2024.

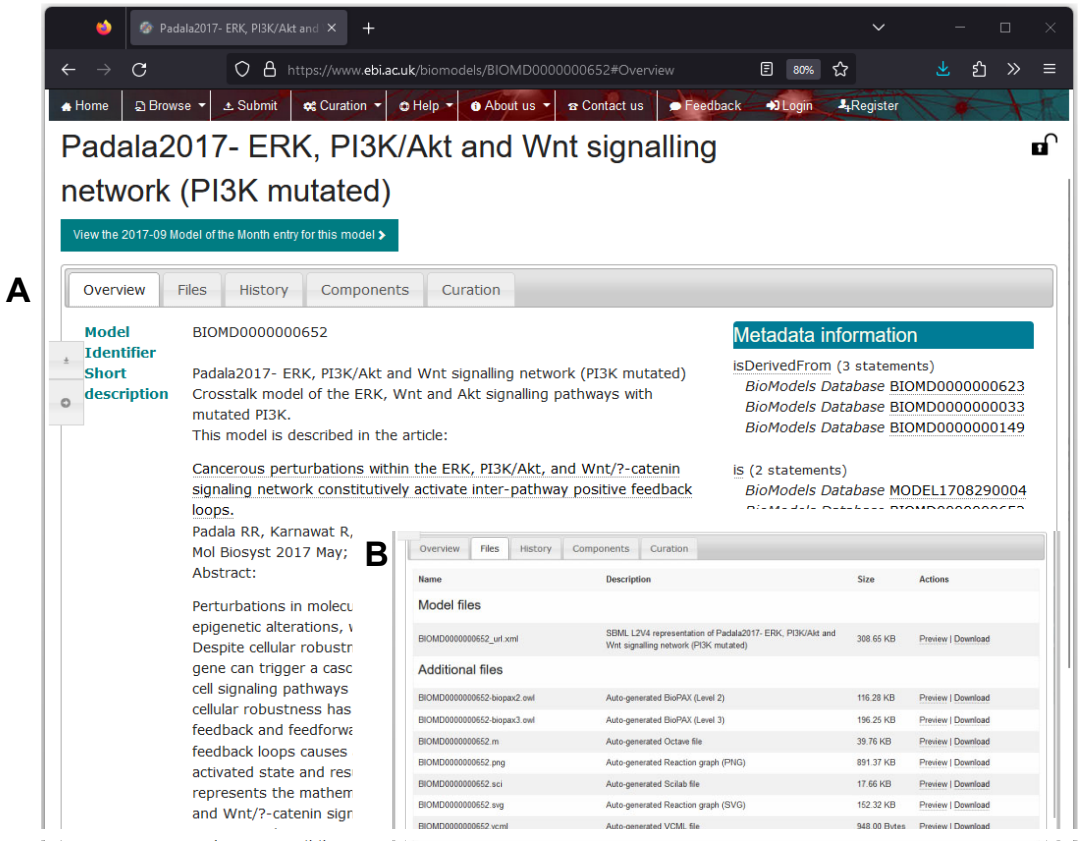


Figure 4.6: The BioModels page of the simulation study by Padala el al. [399] as of July 2024: A) overview of the model, B) list of files uploaded.

provide context about the simulation study. Next, it is demonstrated how an explicit context (encompassing the relationships among all these elements) enables the generation of simulation experiments.

4.3.3 Representing Model Databases as Provenance Graphs

As depicted in Figure 4.6, model databases already contain a wealth of information. However, using the current representation, crucial aspects are not accessible for automatic exploitation and interpretation since it focuses primarily on the final model version. This limitation prevents software for automatic experiment generation from interpreting the context of the study and answering questions about what was the process of developing the model, what analyses were conducted, how and when the model was curated and by whom, and how the model is related to other simulation studies. However, those are crucial for supporting the conduction of simulation experiments during the entire simulation study. Representing the simulation study as a provenance graph may allow software for automatic experiment generation (but also modelers in a manual fashion) to easily navigate and query said information about the entire simulation study. In particular, the standard PROV-DM provides a structured, visual, and machine-accessible approach.

As discussed above in the state of the art, the various forms of documentation can serve different purposes. Thus, combining these documentations promises to provide all the information necessary for reusing the different artifacts and generating simulation experiments. As one example for integrating approaches, this section provides an in-depth discussion of the key considerations when bridging provenance graphs and model databases. It investigates the extent to which provenance pertaining to the overall story of the simulation study is currently included in the

BioModels database, and how this information can be transformed. A subsequent application to a family of cell signaling models shall then demonstrate the advantages of the explicit and accessible provenance graphs regarding automatic experiment generation. Various navigation and querying capabilities are showcased.

When combining provenance graphs and model databases, two cases have to be considered. In the case of new simulation studies, provenance needs to be captured immediately while conducting the simulation study (using one of the techniques listed in Section 4.3.1), and incorporated directly into the model database. Here, the challenge lies in connecting the "model-centric" view with a provenance view, potentially enabling automatic transformation between them. In the case of legacy studies (i.e., entries already existing in the database), a conversion procedure is needed that automatically generates queryable provenance graphs and connects them to provenance of earlier simulation studies. These initial, generated provenance graphs can then be further enriched manually, ideally by the original authors of the study, or automatically by using knowledge extraction techniques [400]. The following focuses on the latter problem: representing existing entries of a model database using PROV-DM. However, investigating the relationship between model databases and provenance will also benefit the former challenge.

There are four main tasks when creating PROV-DM graphs from a model database: 1) recognizing the provenance entities, 2) extracting the metadata, 3) deriving the activities and relationships, and 4) connecting to related studies, with 3) and 4) being the distinguishing features of provenance. In addition, as a preliminary step (0), a suitable data model needs to be defined. The following discusses these steps in detail.

0) Defining a Data Model

The first step before building provenance graphs from entries in a model database is defining a suitable provenance data model. Looking at the PROV-DM ontology for simulation studies described in Section 4.3.1, a translation to the provenance entities Simulation Model (SM), Simulation Experiment (SE), Simulation Data (SD), and the provenance activities Building Simulation Model (BSM) and Analyzing Simulation Model (ASM) can be provided. With respect to further entities and activities, however, the data model needs to be adjusted to fit the data extractable from current model databases, such as BioModels and CoMSES. Since different data used as input cannot be distinguished, such as wet-lab data (previously denoted as WD) or field data, the type Input Data (ID) is introduced to capture various kinds of data. Moreover, since the curation of artifacts is an integral part of various model databases, three additional types are introduced, i.e., the Curation Data (CD) entities, the Curating Simulation Model (C) activities, as well as the Publication (P) entities against which the uploaded artifacts are compared. Furthermore, the conceptual model plays a crucial role in every simulation study. In contrast to the earlier data model, the activity type Building Conceptual Model (BCM) is introduced to capture also the development of the conceptual model entities (from one model version to the other). With respect to the entities of the conceptual model, information in the model databases currently is sparse. Therefore, the following focuses on the qualitative model (QM) to capture the content and context about the modeled system. Entities of type Assumption (A) and Methodology (M) are also briefly discussed. Other entities of the conceptual model, such as research questions, requirements, or simplifications, currently cannot be derived from the information given in model databases. Also, more fine-grained information about the assumptions and methodologies is not available. In addition, calibration and validation activities are not distinguishable from the more general "analyzing" experiments. Explicit annotations would be required to distinguish those.

1) Recognizing the Provenance Entities

Each database entry contains valuable information that can be used to derive the nodes of the provenance graph, and to later fill them with metadata and connect them via activities. In the BioModels database this information is described in different tabs (i.e., Overview, Files, History, Components, and Curation as shown in Figure 4.6).

To identify all provenance entities, one has to go through the different model revisions one by one (given in the History tab) and analyze the information provided, e.g., the uploaded files (via the Files tab). Model databases usually show the latest public version of a simulation study, and previous revisions can be accessed via the menu. However, there may also exist private versions, i.e., versions only visible to the contributors. This is the case when the numbering from version 1 to n (current version) is not consecutive or a version is marked as closed access. The closed (private) versions are ignored in the provenance graphs as no metadata is available for them. For the simulation study by Padala et al., there are three public model revisions available, for which several entities can be created (see Table 4.3). The entities are named according to their entity type (e.g., "SM" for simulation model) and their version number.

Sometimes, files are bundled and uploaded as an archive. In that case, the archive has to be extracted first before the files can be analyzed. E.g., COMBINE archives encoded in the OMEX format [253] are frequently used in the context of BioModels.

Entities of the Conceptual Model: Here the conceptual model will primarily refer to information about the **Qualitative Model**, which describes the contents of the simulation model unambiguously. For each revision, at most one qualitative model entity can be added. This is the case if ontological annotations exist that provide context about the modeled system (e.g., the biological processes of a specific cell signaling pathway), or if files containing a conceptual diagram (e.g., a reaction network given as SVG) exist. Consequently, the Overview and Files tabs of BioModels need to be searched. An **Assumption** may exist, e.g., given by ontological annotations about the modeled organism or cell line. Additionally, an entity of type **Methodology** may exist, i.e., if information about the modeling approach is given. However, in the following case study graphs, assumptions and methodologies will be neglected as they can rarely be filled with further explicit metadata for existing simulation studies in BioModels. As the database is extended in the future, new studies may adopt a broader and partly formalized definition of the conceptual model, such as the one proposed in Section 4.2.3. Then, assumptions or methodologies may be filled with more metadata. And other entities, such as requirements, may become part of the data model used here.

Simulation Model: Simulation models can be recognized by their file extensions (Files tab) or format specified in the metadata (Overview tab). What format the simulation model may be specified in depends on the model database at hand. In the case of BioModels, the simulation models are usually given in SBML [283] or CellML [395]. Currently, each revision must contain exactly one simulation model entity. However, as the database evolves to represent entire simulation studies and the process of conducting them, snapshots of the simulation studies may also involve multiple simulation models. Those can, e.g., represent alternative hypotheses that are analyzed, or submodels to be composed.

Simulation Experiment: A similar approach can be taken for recognizing simulation experiments. Here the list of files is the main source of information. The simulation experiments in BioModels, for instance, are typically provided in COPASI [194] or SED-ML [193] files. As a general rule, a new entity is created for each experiment file found. However, database-specific corner cases have to be considered. For instance, in BioModels often both a COPASI and a SED-ML file are provided for the same experiment. To clearly distinguish which files belong to which entity,

Table 4.3: The provenance entities recognized for the different versions of the model by Padala et al. QM–Qualitative Model, SM–Simulation Model, CD–Curation Data, P–Publication. Adapted from Wilsdorf et al. [302].

Version	Entities	Change
1	SM1, P1, CD1	Initial upload
2	QM2, SM2	Major update and independent curation
3	QM3, SM3	Minor update

the file names or descriptions can help, as they might contain hints such as "COPASI file of experiment xyz" or "SED-ML file of experiment xyz".

Simulation Data and Input Data: The entities of type simulation data and input data can be detected analogously, e.g., by finding CSV files containing raw data, or visualized data as PNGs. Whether a file refers to simulation data or input data currently needs to be derived from the file description. As model databases and provenance get more integrated, the entity types may be annotated explicitly for each file.

Entities referring to the Curation: If curation information exists (e.g., in the Curation tab of BioModels), a curation data entity representing the figures that were reproduced can be created. This entity refers to the output of the curation. In addition, an entity needs to be created for the reference publication (i.e., the curation input) against which the model is curated. Note that in model databases, such as BioModels, currently for each database entry only one (i.e., the latest) curation is shown, even if there have been multiple attempts. Therefore, only one **Curation Data** and one **Publication** entity can be extracted. Nevertheless, those entities have to be assigned the correct version number. In BioModels, for instance, the model version used in the curation may not be the latest uploaded version. Therefore, part of this step is to assess based on which revision the curation was carried out. It can be derived by comparing the timestamp of the curation ("last updated") with the timestamps of the different revisions. The model revision with maximum time stamp less than the curation time stamp will be used. For the Padala et al. simulation study, it can be derived that the independent curation was carried out based on SM2, and thus the curation data and publication entities are added to version 1 (see Table 4.3).

2) Extracting the Metadata

The result of the previous step are a set of entities for each version. These entities can then be refined with metadata. The following section discusses an example mapping of the attributes of provenance entities to the information fields in the current release of BioModels, using examples from the simulation study by Padala et al.

In addition to the discussed attributes, each entity is also assigned an entity name (derived from the entity type and the version number, e.g., "SM3") and a study name (e.g., "Padala et al. (2017)").

Entities of the Conceptual Model: So far, in BioModels the conceptual model is primarily given by the **qualitative model**. Figure 4.7 illustrates how the definition of the qualitative model from Section 4.2.3 can be instantiated with metadata from BioModels. The description of the entity QM3 can be extracted from the *Short description* of the model given in the *Overview* tab. To specify the type of qualitative model, its format, specification tool, and file containing the specification, the *Files* tab is consulted. The URLs to these files can be extracted and added as references to the respective qualitative model entity, and their format can be added as well

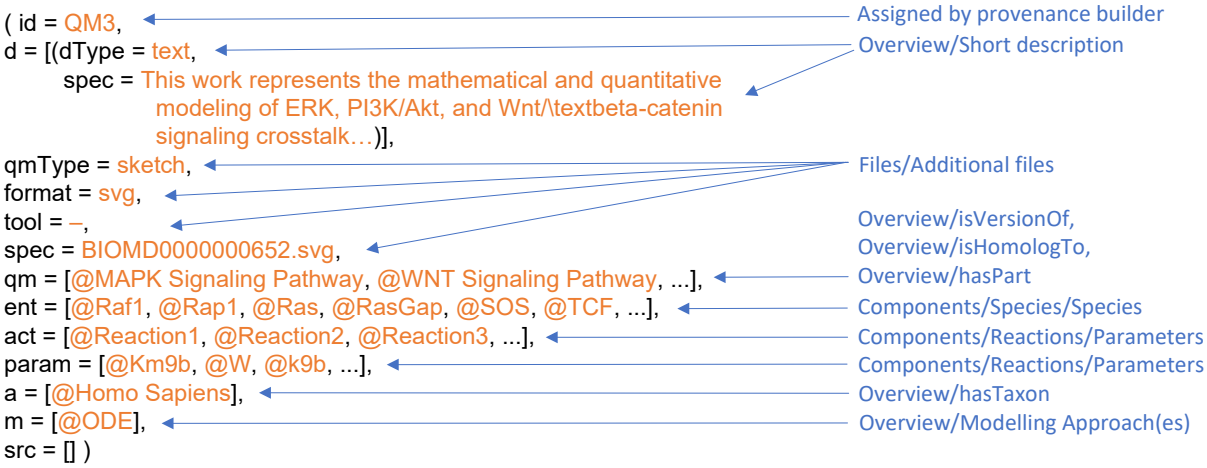


Figure 4.7: Filling the definition of the qualitative model given in Section 4.2.3 with metadata extracted from the BioModels database. Metadata (orange) are exemplified using the study by Padala et al. The exact place in BioModels from where it was extracted is annotated in blue.

(typically in BioModels this will be a reaction diagram showing all the participating model species and the types of reactions between them in PNG, SVG or verbal description in PDF).

To specify the contents explicitly, such as the submodels covered, various annotations at the *Overview* tab can be used. These include annotations regarding which biological processes (Gene Ontology [397]) or which diseases (Human Disease Ontology [401] were modeled. In addition, the qualitative model may be enriched with explicit lists of the model entities, reactions and parameters, all extracted from the *Components* tab of BioModels.

Furthermore, the qualitative model may be linked with explicit assumptions made when modeling, e.g., the organism considered (annotations based on the NCBI Taxonomy [396]), or which cell line the data was based on (BRENDA Tissue Ontology [339]). In BioModels, the ontology tags are given using various qualifiers. For instance, *Overview/hasTaxon* refers to the field named *hasTaxon* in the tab named *Overview* of the BioModels database, which may be linked to the concept *Homo Sapiens* of the NCBI Taxonomy [396].

Additionally, the qualitative model may refer to the modeling methodology to implement it, annotated using the Mathematical Modeling Ontology [323]. For the models in BioModels, these are typically ordinary differential equations (ODE).

Simulation Model: The simulation model entities contain essential information such as a short description, including the abstract of the corresponding scientific publication(s) and instructions for reproducibility (all from the *Overview* tab). In addition, the specification format (typically SBML or CellML) can be stored together with the actual file containing the executable model specification (provided by the *Files* tab). Some models may be given in multiple formats. For instance, specifications in the OWL-based BioPAX exchange format [402] or Scilab [403] may be autogenerated by the Systems Biology Format Converter [404] from an SBML specification (often done in older database entries). Metadata of the simulation model also includes version timestamps and submitter names. For instance, Padala et al.'s latest model submission was on March 21st, 2019 (extracted from the *History* tab).

Simulation Experiment, Simulation Data, Input Data: References, the file formats, and possibly descriptions can be extracted and added to the recognized simulation experiment, input data or simulation data entities, in a similar fashion to the extraction of the simulation model and the qualitative model. In case of the Padala et al. simulation study, no files belonging to

experiments or data can be found. For other models, however, a COPASI or SED-ML file may be included as part of the simulation experiment. What formats are available depends on the database and what formats are currently supported or rather were supported at the time the model was uploaded.

Curation Data: Metadata of the curation data (extracted from the *Curation* tab) includes a short description, the software version used for simulation or plotting, the date and time of the curation, as well as the format and references of the reproduced figures. E.g., for the Padala et al. model Figures 5C and D were reproduced using COPASI 4.19 (Build 140). The information is typically added to the database by the curator in a short comment. In the future, this comment could be further expanded to explicitly annotate which parameter settings were required to successfully reproduce the data or figures from the publication, and to represent this information in a more structured manner.

Publication: With respect to the curation, metadata about the publication to which the simulation results are compared to have to be added. This information is usually given as a URL or DOI in the database entry (*Overview* tab) that references the journal article or conference paper in which the model was published.

3) Deriving the Activities and Dependencies

Provenance graphs are more than just information about individual entities. Once the entities have been identified and filled with metadata, they can be connected by activities and relationships. Table 4.4 provides an overview of the different activity types and the types of entities they use or generate. Those "activity patterns" assist in identifying what type of activity was conducted. Version by version, the available entities are taken into account to derive the necessary activity and dependency types as follows.

1. If conceptual model entities (e.g., a qualitative model) exist, a Building Conceptual Model (BCM) activity is created, with the conceptual model entities as its outputs, and the preceding conceptual model entities (if available) as input.

2. If a simulation model entity exists, a Building Simulation Model (BSM) activity can be created, using the previous simulation model and entities of the conceptual model (if available) as input, and the new simulation model as output.

3. For each simulation experiment entity that exists, an Analyzing Simulation Model (ASM) activity is added, with the simulation model and possibly data as input, and the simulation experiment, and (if available) corresponding simulation data as output.

4. If curation data and publication exist for the current version, a Curating Simulation Model (C) activity is created, taking the publication as input, as well as the simulation model and possibly a simulation experiment and input or simulation data, and generating the curation data.

5. Connecting a curation activity to the right versions of inputs and outputs is particularly tricky. In the special cases, where there is no public model version that can be used as input to the curation, closed versions will be added as proxy entities within the provenance graph. With respect to the outputs, the curation activity may have immediately produced a new version in the database. Thus, all entities of this subsequent version (including qualitative model and simulation model) can directly be added to the provenance graph as outputs of the curation—and the version is skipped by the provenance extractor. To assess if the curation process changed the entities, the timestamp of the revisions and curations

Table 4.4: Types of provenance activities, what inputs they use, and what outputs they generate.

Activity	Used	Generated
Building Conceptual Model	Entities of the Conceptual Model	Entities of the Conceptual Model
Building Simulation Model	Entities of the Conceptual Model, Simulation Model	Simulation Model
Analyzing Simulation Model	Simulation Model, Input Data	Simulation Experiment, Simulation Data
Curating Simulation Model	Publication, Simulation Model, Simulation Experiment, Input Data, Simulation Data	Curation Data, Simulation Model, Simulation Experiment, Entities of the Conceptual Model

can be compared. As a heuristic, the difference between the two dates is used: if it lies approximately within a day, a causal relationship between the curation process and the creation of new (functional and reproducible) versions of the entities can be assumed (e.g., some errors were spotted and fixed immediately). For instance, in the simulation study by Padala et al., the curation activity C1 was based on SM1 and produced the entities of QM2 and SM2, see Figure 4.8.

4) Connecting to Related Simulation Studies

Finally, the database page may provide links to previous models based on which the model at hand was developed. In BioModels, e.g., the related models are referenced via the *isDerivedFrom*-qualifier. These relationships are of particular interest when creating provenance graphs to tell the tale about a family of models [43].

Depending on whether a database entry exists for the related model or only the publication is referenced, either the same procedure as described above is then applied for the related studies recursively, or a single entity is added as proxy for the related model. To connect two studies, a used-relationship is drawn from the first Building Simulation Model activity of the current study to the last version of the related model (or the related model proxy).

In the running example, the study by Padala et al. is related to several other simulation studies. Three of them are explicitly given in BioModels and provenance graphs for those can be created as well, as shown in Figure 4.8. Some of these studies are based on other models too, and thus the provenance graphs can be interlinked further.

4.3.4 Implementation

As proof-of-concept of the presented approach, a web crawler was implemented as a first prototype using Java 8 and the jsoup HTML parser. This tool enables the extraction of provenance information, which can be exported to JSON, a format compatible with provenance editors such as WebProv. It effectively implements the four-step procedure described above for creating a provenance graph from a given BioModels database entry.

The prototype is openly available in a GitLab repository[57]. The repository also contains visual representations of the provenance graphs generated during the case study.

The tool's primary purpose is to facilitate the transformation of legacy simulation studies from the BioModels database into detailed provenance graphs. It also serves in making explicit the relationship between BioModels and provenance graphs of simulation studies in the PROV-DM standard. In the future, those considerations may aid in seamlessly integrating provenance

[57] `https://git.informatik.uni-rostock.de/mosi/biomodels-provenance`, last accessed 19 July, 2024.

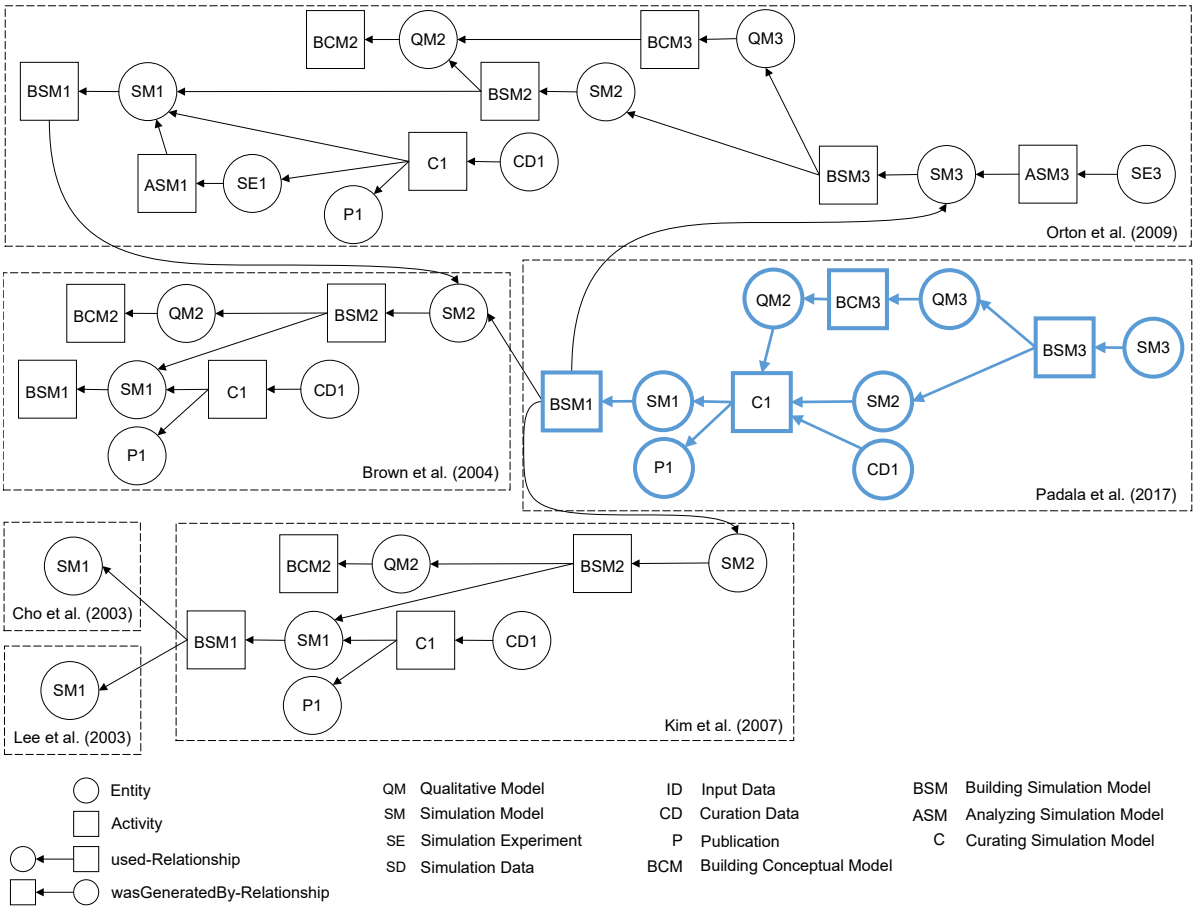


Figure 4.8: Provenance graph(s) created from the BioModels entry BIOMD0000000652, i.e., the Padala et al. study (nodes and edges highlighted in blue) and its related studies.

graphs with BioModels and similar databases. Furthermore, they may facilitate sanity checks for ensuring consistency between provenance graphs and other metadata of newly submitted models.

The created provenance graphs may be visualized using a provenance editior, enabling users to easily enhance the information by adding, e.g., more versions, experiments, or additional details about the conceptual model. The provenance graphs created through this software prototype thus provide an excellent starting point for users seeking to enrich their (BioModels) records.

The prototype was tested to be compatible with the BioModels build of July 2024. Looking forward, the concept would ideally be integrated directly within BioModels. This would not only provide superior support for the users of BioModels but also ensure continuous maintenance of the database-provenance relationship. Furthermore, integrating provenance with other databases would extend the scope of automatic experiment generation to additional information sources.

4.3.5 Exploring and Exploiting the Generated Provenance Graph

By applying PROV-DM and the outlined procedure to the case study and its related cell signaling models, the provenance graph shown in Figure 4.8 can be constructed. The provenance graph formalizes the model building and curation process of the Padala et al. simulation study [399], highlighted in blue, and its related simulation studies, and allows visualizing it. Reading the provenance graph from left to right with the arrows showing back in time, the model building and curation process was as follows: after uploading the simulation model (SM1) to BioModels, it was curated (C1) against the reference publication (P1), which resulted in an auto-generated reaction network as part of the qualitative model (QM2), an updated simulation model (SM2),

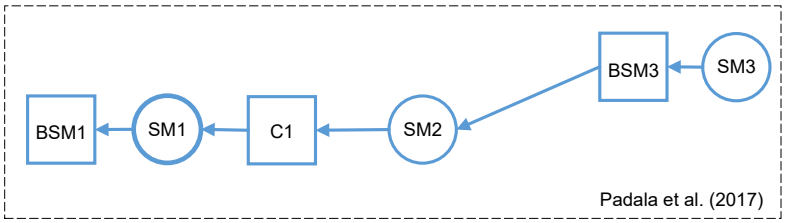


Figure 4.9: Query result showing all the simulation model versions of the Padala et al. study and the activities that produced them.

and curation data (CD1) referring to Figures 5C and D of the publication. The qualitative model was later enriched with various context information about the variables and parameters (BCM3, QM3), and the simulation model was updated (BSM3).

Analyzing a large provenance graph like this, containing multiple connected studies, requires means for zooming into and out of different subgraphs to effectively explore and exploit the given information. Assuming that the provenance is stored in the graph database Neo4j, it can be queried in the Cypher language. In the following, various types of queries are discussed and their role for automatic experiment generation.

To extract all nodes belonging to a particular simulation study, e.g., Padala et al. (see blue nodes and edges in Figure 4.8), the following query can be used:

```
MATCH (s:Study {label: 'Padala et al. (2017)'}), (n {studyId: s.id})
RETURN n
```

This query can be refined into a path query, to zoom further into that study, and extract a path describing all iterations of building the simulation model and showing all associated model versions:

```
MATCH (s:Study {label: 'Padala et al. (2017)'}), (n {definitionId:
  'Simulation Model'})-[]->(a {studyId: s.id})
RETURN n,a
```

Knowing about the model building steps and the status of the model versions (e.g., if they have been validated) is crucial for automatic experiment generation. From the query result (depicted in Figure 4.9) one can see immediately that three versions of the model were published in the model database. A first version, SM1, was built in a Building Simulation Model activity. Then, a second, curated version (SM2) was produced during the Curating Simulation Model activity C1. Later, a third version (SM3) was created.

It is observed that the curation was conducted using the initial version of the simulation model (SM1). This prompts the question of whether the validity of the third model (SM3) remains intact or if it necessitates another round of curation after model building BSM3. If so, simulation experiments would need to be generated for the curation process. The subsequent query provides a means to inquire about all inputs and outputs of curation activity C1:

```
MATCH (s:Study {label: 'Padala et al. (2017)'}), (n)-[]->(a
  {definitionId: 'Curating Simulation Model', studyId: s.id})-[]->(m)
RETURN n,a,m
```

The outcome of the query is presented in Figure 4.10. To further interpret the extracted activities and entities, as well as to determine the actual changes made between different versions, metadata of the entities needs to be considered. In contrast to model databases and their current implementations, provenance offers the advantage of structured metadata organized by entity type. This structural organization facilitates the use of queries to extract specific metadata. The following Cypher query, for instance, returns the software used for reproducing the data.

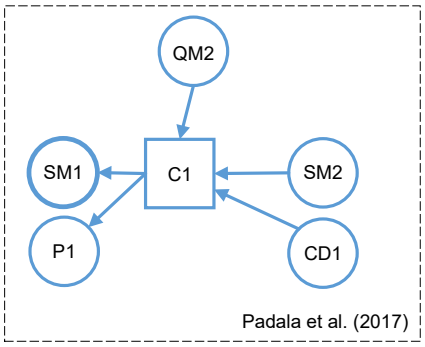


Figure 4.10: Query result showing the curating simulation model activity of the Padala et al. study and all the entities involved.

```
MATCH (s:Study {label: 'Padala et al. (2017)'}), (n {label: 'CD1',
  studyId: s.id})
RETURN n.software
```

The query returns "COPASI 4.19 (Build 140)" as the simulation tool and Matlab R2014b as the tools for obtaining the plots. Thus, when automatically repeating the curation, it is necessary to check for software bindings to these (versions of the) tools or, if required, adapt the code for a more recent version of the software.

Zooming back out, provenance assists in elucidating the relationships between simulation studies, and therefore the bigger picture of a research field, here specifically a family of cell signaling models (including the Wnt signaling pathway). This query aims to identify the different model building activities wherein a simulation model was built by extension or composition of other models:

```
MATCH (n {definitionId: 'Simulation Model'})-[]->(a)-[]->(m
  {definitionId: 'Simulation Model'})
WHERE n.studyId <> m.studyId
RETURN n,a,m
```

Figure 4.11 presents the result of this query. By tracing the relationship-arrows back in time, it becomes evident that the model by Padala et al. was constructed by extension and composition of three other models. Further examination reveals that these models themselves have associations with other models. For instance, it is apparent that both Padala et al. and Orton et al. models were developed upon the model by Brown et al. Additionally, the model by Kim et al. was built by composition of two other models, namely Lee et al. 2003 and Cho et al. 2003, although it should be noted that there are no existing BioModels entries for these two models. Extracting this information about model extension and composition is a crucial step in experiment generation. It helps to identify points in the simulation studies, where cross-checks between models should be conducted, e.g., by validating if requirements or data from the previous studies can still be reproduced or by testing the impact certain inherited assumptions have on the new model.

Last but not least, the provenance graph assists in quickly identifying simulation experiments to automatically reuse and adapt within the same or in a continuing simulation study. Consider, for example, the following query:

```
MATCH (n)-[]->(a {definitionId: 'Analyzing Simulation Model'})-[]->(m)
RETURN n,a,m
```

Its result comprises all analyzing simulation model activities that yielded simulation experiment entities, as shown in Figure 4.12. The query, however, also helps in identifying possibly missing information. Notably, it can be seen that only few simulation experiment entities were made explicit in those studies. For instance, in the study by Orton et al., only two experiment

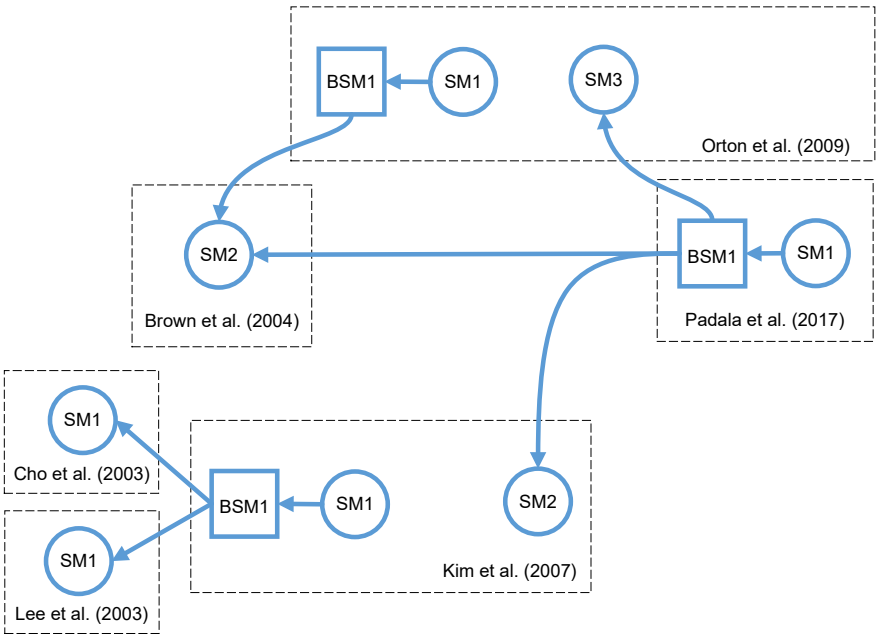


Figure 4.11: Query result showing all simulation model extensions and compositions.

specifications were provided, while in the remaining studies, none were identified by the query—although explicit simulation experiments are integral for the reproducibility and reusability of a simulation study. Thus, in studies where information about experiments is sparse, the modeler could specifically be asked to provide additional information, or one could try to reconstruct information automatically from other sources.

In conclusion, the case study underlined how provenance reveals the intricate relationships in a simulation study. It reveals crucial aspects such as the accessibility of individual entities, the context in which they were produced, the relationships between entities, activities and studies, as well as the presence of rich metadata. Based on graph queries, this information can be extracted and reused for automatic experiment generation. Having the provenance graphs stored as part of online repositories ensures accessibility of the information.

However, in the current state, the provenance graphs that can be generated from BioModels only rarely include the entities of the conceptual model (especially those other than the qualitative model) and simulation experiments. If provenance is eventually captured from the beginning and recorded alongside each activity conducted in the simulation study, more comprehensive provenance graphs can be achieved, such as the one illustrated in Figure 4.13. There, the conceptual model is further resolved into the different research questions, qualitative models, assumptions, and data. Those entity types can then be explicitly linked to the various kinds of simulation experiments, including calibration, validation, and analysis. Moreover, this approach allows for a more fine-grained connection between related simulation studies, e.g., by displaying the various instances in which wet-lab data or an assumption was reused.

Figure 4.12: Query result showing all analyzing simulation model activities conducted in this family of models.

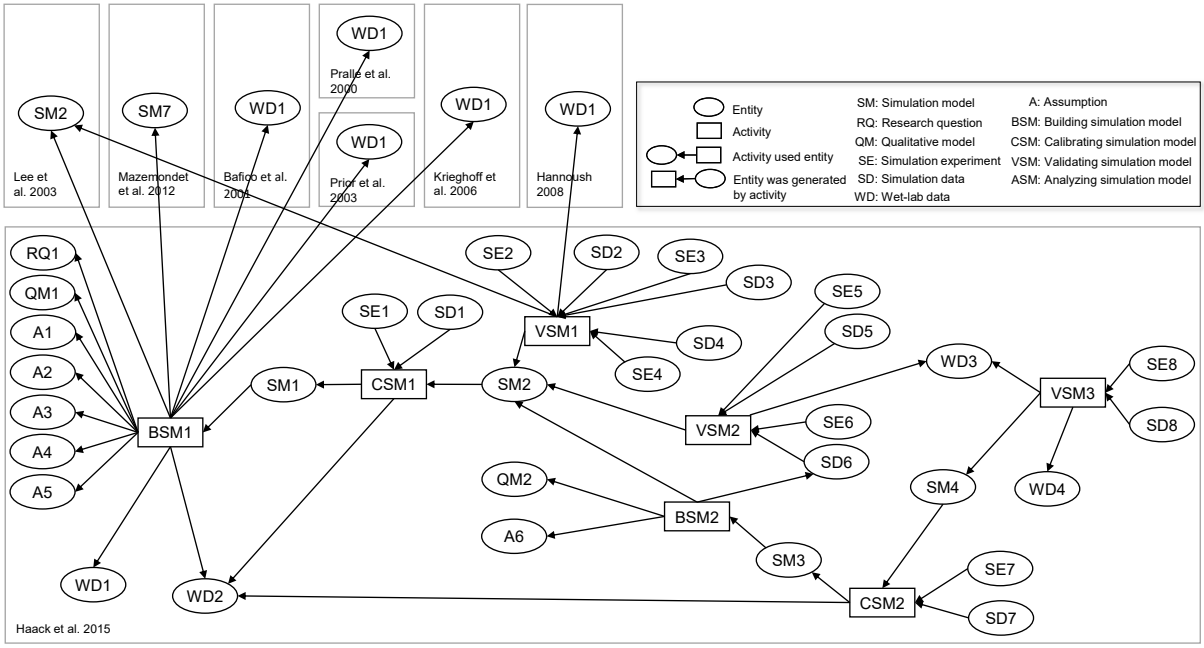


Figure 4.13: Provenance graph of the Wnt signaling simulation study by Haack et al. [31], including a research question, assumptions, qualitative models, wet-lab data, and various simulation experiments and simulation data. Adapted from Budde et al. [43].

4.3.6 Discussion

Documenting the provenance of a simulation study's main products plays a crucial role in improving the understanding of simulation models as well as their reproducibility and reusability. The increasing requirements regarding the reproducibility and reusability of models necessitate explicit, formalized information about how a simulation model was developed and used, including information about the conceptual model and simulation experiments.

Provenance, particularly the PROV-DM standard, has been introduced as a valuable solution. Furthermore, integrating provenance with various other software for supporting and documenting simulation studies has become an essential goal. While integration with workflows, for instance, has been extensively discussed, the potential synergies between provenance and model databases have not yet been explored. However, making provenance information computationally accessible and exploitable (e.g., in a model database) is crucial for developing tools for (partly) automating simulation studies, such as by automatic experiment generation.

Within model databases already an ample collection of simulation models exists, including metainformation and source files. This section was dedicated towards bridging the gap between this information contained in model databases and the PROV-DM provenance standard, which facilitates making the diverse products and their relationships formally explicit. A four-step procedure was devised for creating provenance graphs from entries in a model database—an approach that can benefit databases that want to introduce provenance for existing records and also link it to newly recorded provenance. The steps comprise setting up a data model, recognizing the entities, extracting metainformation about these entities, deriving the activities and relationships, and connecting them to related simulation studies. The concept is illustrated using the BioModels database and various related cell signaling models as a case study.

The resulting PROV-DM representations constitute a valuable addition to model databases like BioModels by making the development, curation, and analysis process of a model transparent. The provenance graphs provide a structured, formalized, and machine-accessible view on the simulation studies, their products, and metadata. Graph queries allow extracting the various entities and activities that played a role in producing the study results and reveal the dependencies

between related studies. The results of the graphs queries can be automatically interpreted and reused to inform and steer the generation of simulation experiments. Furthermore, the generated PROV-DM graphs can be a starting point for collecting more information about all the model building and experimentation steps that have not initially been shared but would be required to fully understand and reuse the various artifacts. This includes making the experiment specifications explicit.

In future work, the various means for capturing, managing and navigating provenance may be enhanced. Those may include new provenance editors or user-friendly query languages. As part of this effort, it will be crucial to customize the provenance data model for the specific model database and its application domains. Currently, not all types of entities (such as Simulation Data, Input Data, or Assumptions of the conceptual model) are universally documented and published for all models. Consequently, they cannot be transformed into provenance graphs by the presented approach. However, with the increasing awareness for reproducibility, publishing these entities will become more common. For larger simulation studies, it could also be interesting to add agents and roles to the provenance graphs to make explicit who contributed during modeling, analysis, or curation.

Naturally, for studies already present in the model database, the generated provenance graphs are limited in scope and focus primarily on the curation history since those models are typically uploaded to the database after they have been published in a journal and therefore only minor changes after publication can be tracked. To get a more comprehensive picture of the simulation study, the curation history (after publication) could be integrated with the development history (before publication). This may be extracted from GitHub commit logs [405]. In contrast, for new simulation studies, it is recommended to capture provenance transparently and right from the beginning to get the entire story of how the different products came into being.

In either scenario, the created provenance views on top of model databases will assist scientists and software tools in better understanding the simulation studies, both manually and automatically. Provenance also enables them to uncover mistakes as well as inconsistent or missing documentation. For example, the provenance representation can help to identify whether or when a curation needs to be repeated after updates on the central entities, such as simulation models or simulation experiments, allowing the respective simulation experiments to be automatically reused and adapted accordingly. Additionally, provenance can help users and software tools to trace back the reasons why models do not produce the same results as in the corresponding publication, ultimately improving the quality and reproducibility of studies.

For now, the provenance graphs are created by a tool developed as proof of concept (see Section 4.3.4). In the future, however, the presented approach can be implemented as an integral part of the various model databases to automatically create provenance graphs for submitted models, and therefore not to burden the modelers with this task. Also, drawing sophisticated conclusions from the graphs can be supported by tools that, e.g., automatically interpret the changes between different versions of the simulation models [406] or other entities. But correctly interpreting changes in computational models is a difficult problem and will need to be investigated further [407].

Although the concept was illustrated and tested specifically for the BioModels database, the general procedure can be transferred to other model databases, e.g., the CoMSES Model Library for computational models in social and ecological sciences [256]. The CoMSES Model Library offers features that are key to the presented approach, including model revisions, peer review, file uploads, and annotations. However, there are some fundamental differences between BioModels and CoMSES that would need to be addressed. First of all, the scope of BioModels is reserved for models from systems biology and biomedicine, typically specified as ODE systems. This well-defined domain provides numerous ontologies and standardized formats that can be exploited to extract provenance entities and metainformation. CoMSES, on the other hand, focuses on agent-based models but is open to a wide range of application fields. While models can be tagged

with keywords, ontologies or structured vocabularies rarely exist for these domains. Furthermore, in agent-based modeling frameworks, such as NetLogo [282], often no clear separation of concerns between simulation model and simulation experiment exists, which limits the types of provenance entities and activities that can be applied. However, with wider adoption of specification languages and frameworks for reproducible simulation experiments like the NLRX R package [192] this gap can be overcome.

Another interesting aspect worth further investigation is the relation between provenance and other forms of documentation. This may include narrative-based reporting guidelines like ODD that are increasingly published as supplements to the simulation models, along with the use of semantic annotations through ontologies. An initial comparison was already made in Section 4.3.1. However, additional work is needed towards effectively integrating these approaches such that the strengths of each can be leveraged.

Finally, provenance graphs serve as an enabling factor for the automatic support of simulation studies by making the central products and their relationships explicit and therefore machine-exploitable. The following Chapter 5 will explore the automatic generation of simulation experiments based on provenance graphs. In this approach, important structural patterns and metadata are identified and queried to extract the necessary information for experiment reuse and adaption.

4.4 Summary

Contextual information about simulation studies is essential as it lays the ground for the entire process of model building, model analysis, and results interpretation. Context includes various artifacts of the conceptual model, which encompass research objectives, requirements, approaches, assumptions, simplifications, qualitative models, as well as data and information sources.

In addition, understanding how these different artifacts relate to each other is crucial context information. This involves details about how the simulation model was successively refined and analyzed and how the conceptual model, along with various data, contributed to this process. Of equal importance are the relations to other simulation studies. Identifying which previous models have been extended or composed, and which data or experiments were reused for calibration or validation, provides valuable insights.

In this chapter, two approaches have been presented to make contextual information explicit such that it can be exploited for improving the process of conducting a simulation study, e.g., by automatic experiment generation. First, a comprehensive and partially formalized definition of the conceptual model has been proposed. This integrated approach allows specifying the contents of various artifacts and establishing references between their data models. An implementation as a MediaWiki facilitates easy specification and browsing of artifacts, illustrated by the conceptual model of the Wnt simulation study (Haack et al., 2015). The case study also emphasizes the role of these artifacts during simulation studies and highlights the importance of explicit and formalized representation. Second, the combination of provenance graphs and model databases was leveraged to tell and share the entire story of a simulation study: PROV-DM graphs relate activities and artifacts even across simulation studies while model databases enable sharing and managing provenance information within a modeling community. A case study of a family of cell signaling models from the BioModels database (Padala et al., 2017 and related models) demonstrates how structured, formal, accessible, and queryable representations of provenance can uncover missing information or inconsistencies in the simulation studies. Moreover, the context in which a model was developed may be explored and exploited semi-automatically to support modelers in conducting their simulation studies.

Although both presented approaches can already be supplemented with tags from domain-specific ontologies, future work should further address domain-specific constraints in making choices about modeling and analysis, and develop further ontologies where necessary. Additionally,

rules about methodological interrelations may have to be specialized for specific domains, as different domains may require particular workflows or state-of-the-art methods for their simulation experiments. Further investigation is needed to explore these differences, particularly concerning the usage of different experiment types, such as optimization, statistical model checking, etc., and to determine the necessary context for making informed decisions about the next steps in building and analyzing the simulation model.

In terms of collecting the vast context information, future work needs to consider and explore new avenues. For instance, natural language processing and process mining techniques may be employed for extracting and organizing relevant information from the scientific literature and code repositories. Advanced approaches for transparent and seamless provenance capture should also be explored. Moreover, conducting large-scale surveys among modeling experts and domain experts may yield valuable insights. Regarding the formalization of context, formal languages for specifying the metadata contained in the artifacts of the conceptual model as well as other provenance entities may be investigated further. For instance, specialized DSLs for specific kinds of requirements may be developed. Last but not least, promoting the adoption of these approaches in various communities is crucial. This can be achieved by showcasing the advantages of making context information explicit, and by developing means for (quantitative) evaluation to provide evidence of their effectiveness. Integration with established platforms, such as various model databases, will also play a key role in facilitating widespread adoption.

The case studies shown in this chapter have already demonstrated the practicality of the approaches for making context of a simulation study explicit (including their artifacts, activities, and relations), revealing their potential for supporting entire simulation studies automatically. Chapter 5 takes this explicit context given by provenance graphs and the conceptual model as well as explicit experiment specifications (see Chapter 3) and combines them in a reuse-and-adapt framework for simulation experiments. This framework allows automatically generating simulation experiments in the context of current and previous activities of a simulation study and their associated artifacts.

5 Generating Simulation Experiments by Reuse and Adaption

5.1 Motivation

Simulation experiments reveal important information about the behavior of a model. Therefore, a wide variety of simulation experiments are conducted during a simulation study [2, 3]. Automatically generating and executing simulation experiments allows simulation studies to be conducted in an easier and more systematic manner. One option lies in the reuse of simulation experiments. In [229, 228], a regression testing approach was proposed in which statistical model checking experiments [159] were reused to check whether the extension or composition of simulation models still exhibits certain behavioral properties. The properties can be interpreted as requirements [223] or hypotheses [219], specifying the expected output behavior of the simulation model that needs to be tested during the development of the simulation model. In the area of cardiac cellular electrophysiology, simulation experiments have been automatically reused to compare different model variants specified in CellML to assess the underlying hypotheses about the mechanisms (reflected in the structure of the models) and their validity [195]. Other approaches focus on the reuse of a simulation experiment's results (outputs) for settings in which experiments are performed repeatedly with the same simulation model, e.g., with various parameter configurations, and aim to increase computational efficiency by avoiding the execution of simulation experiments [408].

In the above approaches, the type of simulation experiment or the kind of simulation model (including the modeling formalisms) have been constrained to support an automatic reuse of simulation experiments in a specific setting. However, various simulation experiments tend to be conducted repeatably with different variants of the simulation model during its development, and thus the repetition of simulation experiments forms a salient feature of the modeling and simulation life cycle. Moreover, having a first simulation experiment already specified gives direction for the steps to come.

To approach the question of how to support the automatic reuse of simulation experiments more generally during simulation studies, necessary ingredients and accessible information sources need to be identified. A prerequisite for the reuse of simulation experiments is a clear separation of concerns between model, simulator, and simulation experiment. In addition, simulation experiments need to be explicitly specified to be accessible and reusable [212]. Over the last two decades, various approaches have been developed that allow an explicit specification of simulation experiments. To those belong model-based approaches such as [222], domain-specific languages such as SESSL [36], or standardized formats such as SED-ML [193]. Only if simulation experiments are explicitly specified, they can be automatically interpreted. Their interpretation is facilitated by schemas [221] and metamodels [17] for the different types of experiments, possibly complemented by ontologies about their various roles [223], designs [120], and methods [238].

Also the past contains valuable information that can be exploited in a variation of Santayana's phrase "Those who cannot remember the past are condemned to repeat it": Those who can remember and interpret the past can effectively plan the steps ahead, which might include, in the case of simulation studies, a deliberate repetition of steps. Provenance information about simulation models reveals crucial information about a simulation model's past in terms of how a model has been developed and executed. This includes information sources as well as activities, such as the conduction of simulation experiments that contributed to its development [28]. Provenance

information may be used to relate information sources, activities, and generated entities, within and beyond individual simulation models thus forming entire families of models [43].

In related work, provenance information from simulation studies has been exploited but the motivation behind it was different. For instance, provenance about executed simulation experiments was used to reduce the computational load of simulation platforms by avoiding duplicate simulations [409]. Other approaches exploited provenance about previous workflow runs to optimize workflow execution time by allocating the necessary computing resources [410]. In the following, the question will be pursued of how to automatically detect new experiments to be reused based on what has been done before. The reuse of simulation experiments refers to the reuse of a simulation experiment specification, which is then adapted and executed.

As the central building block of the approach, typical patterns that can be observed in the provenance graph's of simulation studies are defined and associated with semantics. Based on the patterns, rules are specified that automatically identify experiments to reuse, and then adapt, generate, and execute a new experiment. Updates of the provenance graph function as triggers to this process. The approach is implemented as the open-source *Reuse and Adapt framework for Simulation Experiments* (RASE). The utility of the framework is demonstrated in two simulation studies from demography and cell biology, respectively: one aimed at developing a simulation model to study the impact of information flow on migration, the other aimed at revealing crucial mechanisms of a central signaling pathway. The case studies show that simulation experiments as well as other provenance entities can be effectively reused and exploited for automatically generating a simulation experiment. Finally, the special case in which only information about context is reused without an earlier experiment as blueprint is discussed.

This chapter is based on the publication [237], which introduced the RASE framework. The ideas from [19] about generating simulation experiments from scratch have been incorporated in Section 5.8. Other publications that are loosely related to the work of this chapter are [220, 43, 252].

The approach presented in this chapter now aims to assemble the various concepts for making simulation experiments and their context explicit, which were discussed in Chapters 3 and 4. The provenance graphs are the central methodology for expressing knowledge about the relationship between different kinds of activities and available entities in the form of patterns and rules. Section 4.3.1 discussed the idea of provenance and described a provenance model for simulation studies. In addition, several approaches for capturing provenance graphs automatically during a simulation study were discussed, as well as means for representing and distributing provenance graphs via model databases (Section 4.3.3). The experiment metamodels and the MDE pipeline for simulation experiments presented in Chapter 3 are now used within RASE for facilitating the code generation and adaption. Parsers and generators for domain-specific languages are required for handling existing simulation experiment specifications in a simulation backend of the user's choice. In the adaption of simulation experiments, ontologies are essential in interpreting changes and disambiguating the information given. Ontologies are also crucial for choosing and parameterizing the right method for a simulation experiment in alignment with literature and common practices (see excursus in Appendix B). Currently, the step of selecting the right method is not considered in this reuse-based approach, but in Section 5.8 the challenges of generating a simulation experiment from scratch and the integration of ontologies are discussed. And lastly but equally importantly, provenance information with rich and partly formalized metadata (e.g., based on the broader definition of the conceptual model presented in Section 4.2) is crucial in automatically reusing simulation experiments, determining how the next experiment looks like, and performing the necessary adaptations automatically. Even more detailed information is required if simulation experiments shall be generated from scratch.

5.2 State of the Art

Due to the ongoing reproducibility crisis [243], changes in how simulation studies are conducted (and thus how simulation experiments are carried out) have been required. Therefore, in recent years, simulation researchers investigated the reuse or generation of simulation experiments or parts thereof. Table 5.1 summarizes the state of the art on reusing or generating simulation experiments with regards to their overall objective, the central methodology applied, the context information used, the experiment types and the simulation tools supported, and the scope of the automation (i.e., how the generation process is initiated, and whether the approach can be used within or across simulation studies). These are compared to the features of the approach presented in this chapter, i.e., the Reuse and Adapt framework for Simulation Experiments (RASE).

Hillston et al. [411] and Hillston [241] describe the Experimenter tool as part of the Integrated Modelling Support Environment (IMSE). It is a tool for creating and running experiments for performance evaluation and validation of systems. It features both a textual and a graphical interface that guide the user in planning their experiments. The user can express which workloads to simulate (input) and what results they are interested in (output), and how the model is to be executed (control). The Experimenter tool allows for the creation of an experimental plan that clearly defines the necessary model executions and subsequent analysis of results. This plan can consist of multiple experimental frames [240], which are small subexperiments with a common objective (e.g., individual model executions within a parameter scan). There are also generic, parameterizable plans (templates) that can be tailored to specific needs. This is related to the concept of using schemas or metamodels to represent the ingredients of an experiment (see discussion in Chapter 3). Moreover, the planning tool can support experimentation independently of modeling paradigms and formalims. The experimental plan executor can then automatically execute the plan, invoking the paradigm-dependent execution engine. As context for the execution, explicitly defined dependencies between parameters and variables of the simulation model are used to constrain the search space in the generated parameter scan.

Work by Birta and Özmizrak discusses the use of pre-structured knowledge for the validation of simulation models [412]. The objective of their theoretical framework is the generation of efficient experiment designs for checking behavioral requirements. The approach uses a "validation knowledge base" that contains the expected behaviors of the simulation model. These may be quantitative or qualitative requirements.

The model-driven engineering (MDE) based approach by Teran-Somohano et al. [222] targets the generation of factorial experiment designs for single simulation experiments anywhere in the life cycle of a simulation study. The experiments are generated based on user inputs submitted via a GUI, without considering provenance or other context information. Repast is used as the modeling and simulation tool. However, due to the MDE-based design, other tools could be interfaced.

Yilmaz et al. demonstrate the potential of MDE for generating experiment designs for hypothesis testing as part of a goal-hypothesis-experiment framework [227]. The framework does not automatically target specific steps in the modeling and simulation life cycle; instead, and the experiment generation has to be triggered by the user by inputting formal hypotheses (derived from the overall research goals). Consequently, the use of a domain-specific language (DSL) for the specification of hypotheses is central in this approach. As the framework is merely theoretical, presently no concrete implementation exists.

In a similar approach, Lorig also uses a DSL for hypothesis specification, and passes the formal specifications as context information to the experiment generation [219]. From the hypotheses, hypothesis tests are generated in NetLogo but other agent-based modeling and simulation tools are discussed as well.

The approach by Peng et al. targets the reuse of statistical model checking experiments

after model extension [229] and composition [228] to check whether the extended or composed model still exhibits the expected behavior. In both cases, experiment specifications need to be adapted to the extended/composed model, whereas the latter case also requires syntactic and semantic composition of formal hypotheses. Hypotheses are formalized as logic formulas in Metric Interval Temporal Logic (MITL) [167]; simulation experiments are generated in the specification language SESSL. The approach bears similarities to RASE, however, the reuse and adaptation must be triggered by a user, providing the original model(s) and experiment(s), adaptation information (i.e., a list of changes to be applied), result expectation (logic formulas) and the extended/composed model as context.

The cardiac electrophysiology web lab, designed by Cooper et al., targets the comparison and validation of simulation models [195]. Similar to RASE, all models, experiments, and data are stored in a database. When a user uploads a completed model via the web interface, it can be automatically cross-checked with other models by running all available experiments on the new model. A domain ontology assists in selecting the relevant models and experiments from the database. Due to the limitation to the domain of cardiac electrophysiology, and the use of well-defined formats and naming conventions, a high degree of automation can be achieved. However, the process of developing a model, including the various iterations of model building, calibration, and validation, is not supported.

Ruscheinski et al. [220] also do not incorporate the process of conducting a simulation study, but aim to generate simulation experiments from scratch using a static documentation. Their approach is based on templates defined in a template language. Variables of the templates can be filled with information provided in the model documentation. Specifically, formally specified requirements in MITL and parameter tables from within the same simulation study are exploited. To generate an experiment, user inputs are passed to the template engine together with inputs extracted from the user-provided documentation. The template engine fills in the SESSL code given by the templates accordingly. Supported experiment types, thus far, are statistical model checking and local sensitivity analysis.

The artifact-based workflow by Ruscheinski et al. can guide users through the various steps of a simulation study, and to re-visiting the calibration and validation stages of the simulation model if certain milestones are achieved or invalidated [223]. Thus, artifacts, their information models and milestones serve as context, including the conceptual model, requirements, simulation models, simulation experiments [223], and data [52]. When the validation stage is visited, all experiments with role validation are (re-)executed in the workflow. The concept of toolboxes enables the flexible use of different simulation tools and experiment types inside the workflow. However, the experiments are not automatically specified, adapted and executed by the workflow but information about artifacts needs to be submitted by the user through the workflow GUI. In the future, the artifact-based workflow may be combined with RASE: the artifact-based workflow would function as a provenance recording tool, and RASE would automatically evaluate the provenance information to generate the next simulation experiment.

In contrast to the existing research, RASE provides a provenance-based mechanism for automatically reusing, adapting, and executing simulation experiments during the simulation study. Exploiting provenance information of the simulation study given in the provenance standard PROV-DM allows RASE to reason about the various model building and experimentation activities, the entities used and produced, and the precise relationships between them. In the inference rules, activities can serve as triggers to the experiment generation, and for automatically selecting suitable simulation experiments to reuse. Thus, the approach is not restricted to specific settings, i.e., it uses provenance patterns to recognize whether a calibration, validation, or analysis experiment is needed. In addition, it can cover any experiment type, be it factorial scan, statistical model checking or optimization. Furthermore, provenance contains valuable information about the context of the simulation experiments that allows for an automatic adaption of the reused experiment specifications. Provenance used by RASE includes at least the research questions,

Table 5.1: Comparison of related work on automatically generating and executing simulation experiments. n.a. = not applicable. Extended from [237].

Publication	Objective	Central Methodology	Used Context	Experiment Types	Supported Formats, Tools	Generation Trigger	Across Studies
Hillston et al. (1991), Hillston (1995)	Creation and execution of experiments	Experimental frames, Templates	Dependencies between parameters and variables	Performance evaluation and validation	Flexible	User input	no
Birta and Özmizrak (1996)	Generation of efficient designs for validation experiments	Knowledge base of expected behaviors	Requirements	Requirement checking	n.a.	n.a.	no
Teran-Somohano et al. (2015)	Generation of experiment designs	Model-driven engineering	n.a.	Factorial designs	Repast	User input	no
Yilmaz et al. (2016)	Generation of experiment designs	Model-driven engineering, Hypothesis language	Hypotheses	Hypothesis testing	n.a.	User input	no
Lorig (2019)	Generation of experiment designs	Hypothesis language	Hypotheses	Hypothesis testing	NetLogo	User input	no
Cooper et al. (2016)	Comparison and validation of electrophysiology models	Online database	Models, Experiments, Data, Domain ontology	Flexible	CellML, SED-ML	User input	yes
Peng et al. (2016, 2017)	Reuse and adaption of simulation experiments for model extension and composition	Composition of experiments and hypotheses, Hypothesis language	Models, Experiments, Hypotheses, List of changes	Statistical model checking	SESSL, MITL	User input	yes
Ruscheinski et al. (2018)	Generation of experiments from scratch	Templates, Model Documentation	Requirements, Parameter Tables	Statistical Model Checking, Local sensitivity analysis	SESSL, MITL	User input	no
Ruscheinski et al. (2019, 2022)	Guidance for re-running validation and calibration experiments when milestones are achieved or invalidated	Artifact-based workflow	Conceptual model (Requirements), Models, Experiments, Data	Flexible	Flexible	User input	no
RASE (2023)	Situation-specific reuse, adaption, and execution of simulation experiments for calibration, validation, and analysis	Inference rules, PROV-DM patterns, Model-driven engineering	Provenance graphs showing the Relationships between various Entities (incl. Conceptual model, Models, Experiments, Data), Activities, and Studies	Flexible	Flexible	Automatic recognition of specific activities	yes

assumptions, requirements, qualitative models, simulation models, simulation experiments, and data. Other entities may be added on demand. In contrast to Peng et al., where the adaption information needs to be provided by the user specifically for each experiment generation, this information can now be automatically extracted from the provenance of the simulation study. Integrating this with an MDE approach for generating and translating simulation experiments allows RASE to flexibly support a variety of experiment types and a variety of modeling and simulation tools. In contrast to Ruscheinski et al. [223], the built-in MDE pipeline also facilitates automatic reuse across simulation studies, even if they use different tooling, as experiment specifications can be translated automatically.

5.3 Typical Reuse Scenarios in Simulation Studies

According to the International Vocabulary for Metrology [413], there are different levels of reuse. These definitions have also been adopted by the ACM (Association for Computing Machinery), in agreement with the National Information Standards Organization (NISO), for their artifact reviewing process[58].

The first level is *repeatability*, which refers to the repetition of an experiment within the same simulation study, typically by the same investigators with the same experimental setup. The second level is *reproducibility*, i.e., repeating an experiment outside the simulation study, but with the same experimental setup. The third level is *replicability*, i.e., enabling the reuse of a simulation experiment in a different study with a different experimental setup. This includes, e.g., using another language or tool to specify and run the experiment, or using a different set of validation data. Finally, ACM also recognizes that simulation studies should be "documented and well-structured to the extent that reuse and repurposing is facilitated". Therefore, *repurposing* can be seen as a fourth level of reuse, where the products of a simulation study are reused outside of the original context with a new intention.

The following subsections describe various scenarios in which the proposed framework, RASE, could be of assistance. Although all of these scenarios can be associated with one or more of the four reuse levels, the distinction is not so clear and depends on the given context, as in the case of re-validation (see Section 5.3.1). The re-validation scenario could be interpreted either as a) repeating if the experiment is reused in the same study and the input and configuration of the experiment were not affected by the last model refinement, b) reproducing if the experiment is reused in another study but with the same configuration, c) replicating if, e.g., new data are used as a reference for the validation, or d) neither of those if different experimental setups are assessed inside the same simulation study. Furthermore, in practice these terms are often used interchangeably.

Therefore, this chapter will not use the above definitions but merely refer to either the reuse of simulation experiments and other products *within* the same simulation study, or the reuse *across* studies, i.e., the reused entities are taken from another related simulation study, presuming that the provenance graphs of the two studies are connected. In both cases, simulation experiments are repeated and data are reproduced with regard to the current situation and the context of the reused entities. Note also that the following scenarios apply to simulation studies where the development of a valid simulation model is the focus. In these kinds of studies, the reuse of simulation experiments is typically evoked by the creation of a new simulation model or simulation model version.

As discussed in Section 2.2.1, there is some literature that argues for distinguishing the role of a simulation experiment between calibration, validation, and analysis. All of these may pertain to various types of simulation experiments, e.g., a sensitivity analysis experiment for analyzing the simulation model (cf. Section 2.2.3). The approach pursued in RASE and the supported

[58] `https://www.acm.org/publications/policies/artifact-review-and-badging-current`, last accessed 19 July, 2024.

scenarios will be structured accordingly, referring to the reuse of simulation experiments for calibration, validation, and analysis.

5.3.1 Repeated Model Validation after Model Extension and Composition

Validation is an important task in the modeling and simulation life cycle. It is the process of substantiating whether the model behaves consistently with the modeler's expectations, e.g., regarding data that has been measured in the real system [100]. However, the model development is usually not completed after the first validation. Further model features are added, and the modeler moves again through the phases of the modeling and simulation life cycle. Making changes to the simulation model then requires re-validating it, as manifested in the artifact-based workflow by Ruscheinski et al. [223] and the reuse of simulation experiments for model extension by Peng et al. [229]. In software engineering, successive validation is known as regression testing: "Regression testing is performed between two different versions of software in order to provide confidence that the newly introduced features of the [system under test] do not interfere with the existing features" [414]. For the scientific community, also test-driven model validation procedures have been proposed [415].

A special case of this regression testing occurs in model composition. Frequently, models are not built from scratch but are created by composition of existing models. The composed models may have been developed as separate modules during the same simulation study or may originate from different, previously conducted simulations studies. For instance, during the COVID-19 crisis, the composition of modules previously validated against independent data enabled the rapid development of a model to inform policy decisions in Austria [416]. Once two or more (validated) models have been merged, it should be evaluated if everything still behaves as expected [228]. Thus, validation experiments that were conducted with the individual models are typically repeated with the composed model. This process applies independently of the used tools and formats, and for true composition as well as model fusion. In composition, models are constructed from interacting submodels, whereas in fusion, models are combined irreversibly and the identities of the submodels vanish [417]. In multi-disciplinary studies, the composition might also refer to a co-simulation [418], where after orchestration of the simulation units, requirement checks are repeated on the global level before the partial solutions are combined.

5.3.2 Repeated Model Calibration

Calibration, also called model fitting, is the process of finding a parameterization of the model that can reproduce observed behavior of the real system [100]. While calibration and validation both typically relate to real data, the conclusions drawn from them are essentially different. Whereas calibration refers to adjusting the input parameters such that the resulting agreement of the model output with a chosen set of experimental data is maximized, the goal of validation is to establish confidence in the model predictions [151]. Therefore, if previous calibration experiments exist in a simulation study, they should be repeated before the validation experiments. Consequently, when a new model version is produced, calibration experiments are repeated. Only if the calibration was successful, can the validation be attempted. If the model cannot be calibrated such that it reproduces the data with sufficient accuracy, further model revisions are necessary.

5.3.3 Repeated Model Analysis

In simulation studies, many experiments are conducted that do not serve as validation or calibration. Nevertheless, they still reveal important information about the model, e.g., via parameter scans, optimization, sensitivity analysis, perturbation analysis, or time course analysis [43]. It could be argued that all experiments contribute to the validation of the simulation model as they increase the trust that the right model has been built. In the context of this work, it is assumed

that validation experiments are experiments that are distinguished as such by the modelers, and whose outcome can be evaluated as a *success* or *failure*, see also [223]. Especially sensitivity analysis is increasingly becoming an integral part of simulation studies as it allows quantifying how the parameters contribute to the uncertainty in the model output [419]. It is becoming good practice to attach each model version with uncertainty and sensitivity information. Therefore, automatically reusing and repeating sensitivity analysis experiments after each major model version is instrumental in enhancing the quality of simulation models.

5.3.4 Cross-Validation with Related Simulation Studies

Cross-validation, or model alignment, is the process of comparing a simulation model with another, independently developed model [420]. In particular, models that deal with a similar research question should be able to reproduce each other's results. By comparing (i.e., validating) simulation models with other, already calibrated and validated models, a "domain validity" can be achieved and overall confidence in the models can be established. To facilitate cross-validation, simulation experiments conducted with related, validated models should be reused and adapted in the validation activities of the current simulation study.

5.3.5 Comparison of Alternative Implementations

Usually, there is not just one way of modeling a system. From the same conceptual model different computerized models can be built. For example, the same model can be simulated using discrete event simulation (based on continuous-time Markov chains) or using system dynamics (based on ordinary differential equations), and with or without spatial features. Comparing alternative modeling and simulation approaches is crucial, e.g., to uncover discrepancies in simulation results, or to find a more efficient implementation in terms of simulation runtime.

The comparison is done by reusing and reapplying simulation experiments that were conducted with the other model implementation. Sometimes this comparison is done as part of the same simulation study, e.g., when switching to a different platform to improve performance. However, the re-implementation of a previous model could also be the primary goal of a simulation study to gain a better understanding and confidence in the results [421]. In this case, simulation experiments have to be reused across simulation studies.

5.3.6 Synchronization of Concurrently Developed Models

In large simulation studies, often numerous models, e.g., candidate models or submodels, are developed concurrently. This can be either the work of a single modeler or multiple modelers in a collaborative simulation study. In collaborative workflows, the sharing and reusing of data and other products between peers is crucial to keep the work on the different branches synchronized [422]. For instance, when one modeler completes a calibration in one branch, this could trigger new experiments for the other submodels based on the calibration results.

5.4 Reuse and Adapt Framework for Simulation Experiments

To automate the reuse of simulation experiments as in the scenarios described above, the Reuse and Adapt framework for Simulation Experiments (RASE) is proposed. The framework presents a provenance-based mechanism for reusing simulation experiments either within or across simulation studies. It exploits the observation that certain user activities of the modeling and simulation life cycle produce characteristic patterns in the provenance graphs. Based on such patterns, the production rules of a graph transformation system can be constructed. In the following, first the architecture of the framework is introduced. Next, the provenance graph transformation system is defined, which allows producing new experiment activities based on patterns of the last

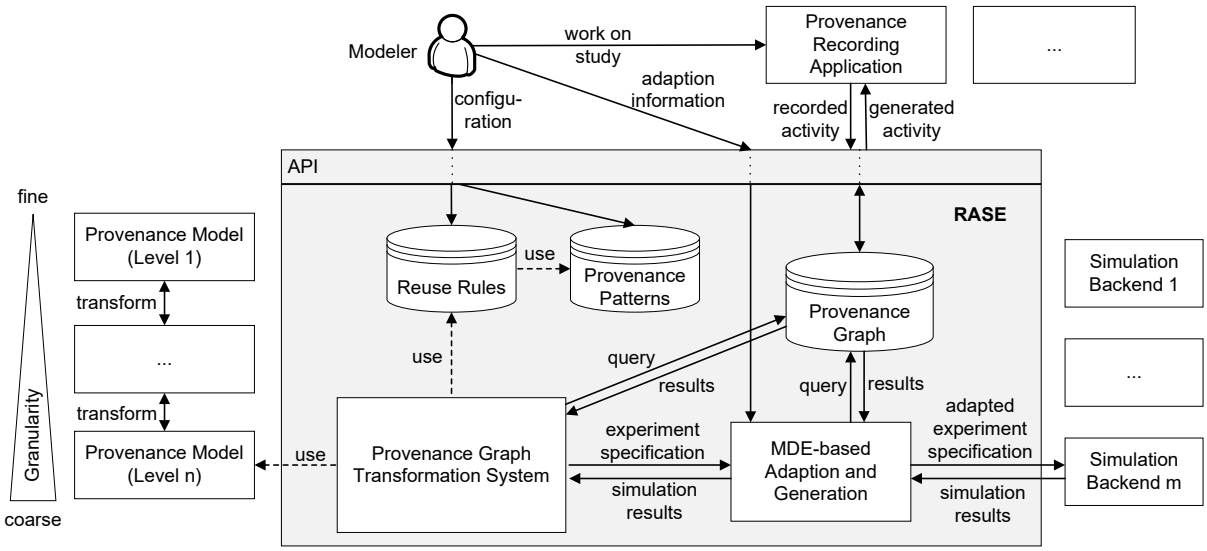


Figure 5.1: Overview of the reuse and adapt framework for simulation experiments (RASE). Adapted from [237].

modeling activity and previously conducted simulation experiments. The section concludes with the question of how to adapt simulation experiment specifications by taking context information about the latest model building activity into account.

5.4.1 Architecture

Figure 5.1 provides an overview of the framework's architecture. The entry point is the application programming interface (API) that allows submitting provenance from a variety of applications, ideally on-the-fly while the modeler works on the simulation study. The provenance recording applications may be stand-alone GUIs or workflow systems that capture provenance while the users run through a number of workflow stages. The recorded provenance is stored in a graph database. RASE observes the database and recognizes when a new activity is committed. Each newly added activity initiates the evaluation of the provenance graph transformation rules. When a rule can be applied, and thus an experiment shall be reused for a new simulation model, the graph transformation system coordinates with the adaption and generation component. This component is required to transform the old experiment specification to the context of the new simulation model. The adaption is based on a model-driven engineering (MDE) approach that uses metamodels for automatically identifying and accessing the parts of experiment specifications that have to be adapted based on new provenance information. In some cases, however, it might be necessary to interact with the user during this step as they may carry relevant yet implicit knowledge about the simulation study. For example, via the API the modeler may be asked to specify a requirement in a formal manner or to review the adapted experiment design. After the adaptions, the MDE component generates experiment code for a specific modeling and simulation tool called backend. The code can be automatically executed using a backend binding. When the experiment terminates, the simulation results are returned to the graph transformation component and added to the provenance graph, as they contain valuable new information about the simulation study that needs to be documented. The new provenance created by the rule evaluation can be displayed to the user via the API.

Note that for now, the framework expects a provenance model that provides a macro-level view on the provenance (see also discussion in Section 4.3.1). This allows it to efficiently interpret the intention behind an activity (e.g., calibration vs. validation). If provenance is recorded on a more fine-grained level, aggregation methods are used to obtain the required provenance view.

However, in the future, this framework could work on arbitrary provenance views if adequate patterns and rules were specified. The configuration of the rules and patterns can be done by any user of the framework via the API.

5.4.2 Experiment Reuse by Provenance Graph Transformation Rules

The framework is driven by rules made up of provenance patterns, which form a graph transformation system that, based on a given provenance graph, extends this graph with new simulation experiments.

Provenance Patterns in Simulation Studies

Provenance patterns are the central building block for RASE. A provenance pattern defines an interesting subgraph of a provenance graph in terms of the contained types of nodes and relationships as well as metadata. Associating provenance patterns with semantics of modeling and simulation enables RASE to interpret what happened during the simulation study. In particular, patterns are used in four ways.

1. *Trigger patterns* initiate the reuse, adaption, and execution of certain experiments. In the dissertation at hand, the focus is on experiment reuse after changes in a simulation model have been detected. A trigger pattern, thus, always denotes a provenance activity that produces a simulation model. Once a trigger pattern matches in the current provenance graph, all the involved provenance nodes and relationships are retrieved as context for the experiment generation. Note that, especially if more fine-grained provenance information is available, other triggers that do not produce a simulation model might be possible (e.g., the successful or failed execution of an experiment or the specification of a behavioral requirement).

2. *Experiment patterns* are used to identify previous simulation experiments. They describe either a provenance activity with a specific role w.r.t. experimentation, e.g., model calibration, validation, or analysis [43], or an activity that produces an experiment of a specific type, such as sensitivity analysis or statistical model checking. If a pattern matches, the respective nodes and edges are retrieved as context for the experiment generation.

3. Eventually, from the retrieved provenance information a new activity as well as new entities for the adapted simulation experiment and its results will be generated. For this, experiment patterns are used as *blueprints* for creating the respective nodes in the provenance graph and connecting everything correctly.

4. *Condition patterns* represent the relationship between the trigger patterns, experiment patterns, and the rest of the provenance graph. They are particularly important when expressing rules for complex use cases that need to incorporate further context of the activities.

In the following, various types of trigger patterns, experiment patterns, and condition patterns are predefined based on experiences with simulation studies (from cell biology) where the development of a valid model is in focus and thus multiple model iterations are produced [43]. These patterns cover the scenarios described above in Section 5.3 and are illustrated in Figure 5.2. However, this list is not exhaustive and there may be special use cases and patterns from other domains. The modular architecture of the framework allows users to add custom patterns and to use them in defining new rules.

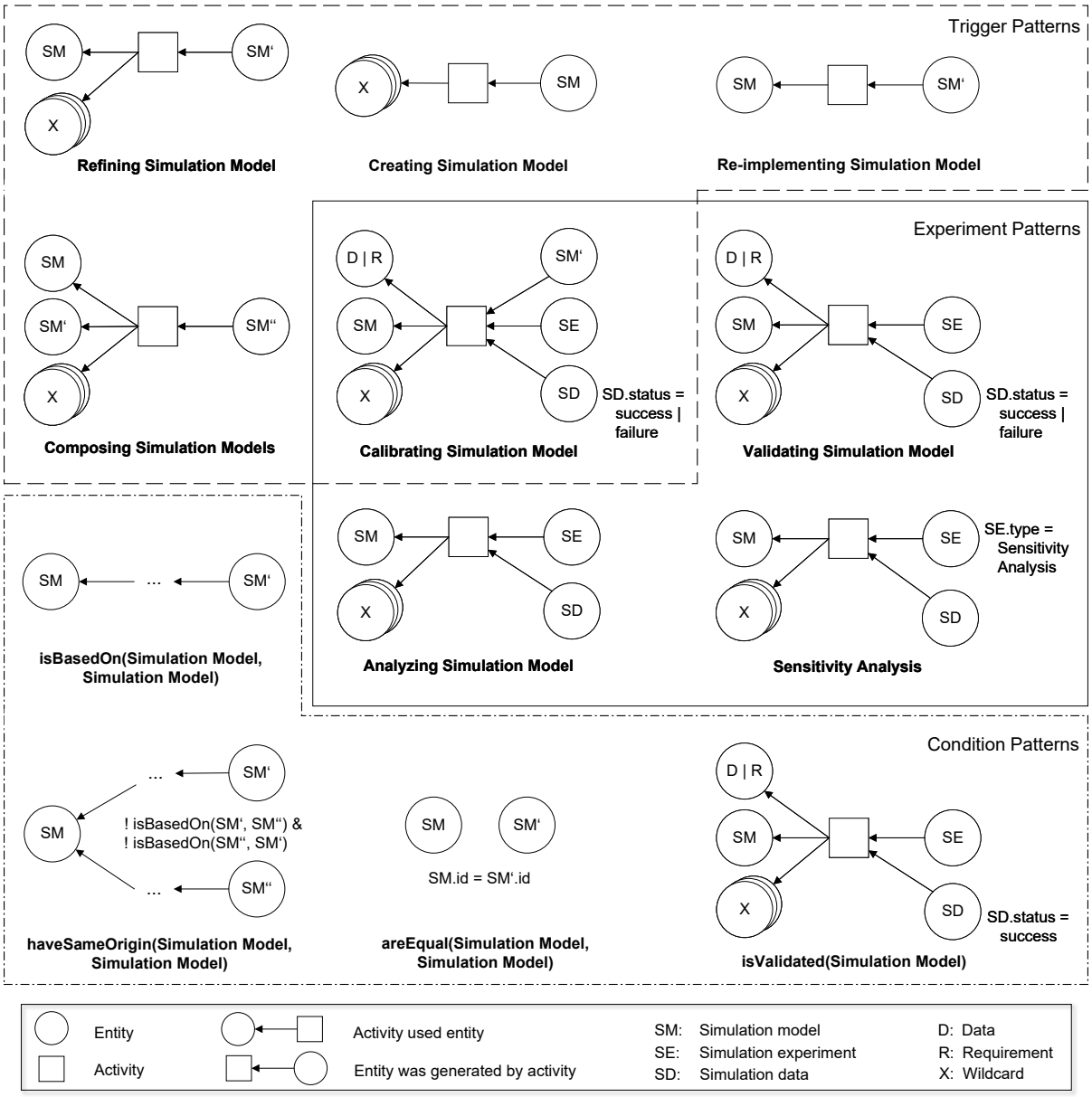


Figure 5.2: Provenance patterns for interpreting changes in the simulation models (trigger patterns), finding suitable simulation experiments to reuse (experiment patterns), and expressing the relationship between trigger pattern, experiment pattern, and the rest of the provenance graph (condition patterns). The patterns show what entities are required as inputs and outputs for a certain type of activity. The multi-entities (X) are used as wildcards to capture all remaining inputs, however, they must not contain entities of the type simulation model (SM). Different entities of the same type are denoted by the prime symbol ('). Adapted from [237].

"Refining Simulation Model" Pattern: Model refinement or extension is the typical model-building step during a simulation study. It involves a single simulation model and produces a new one. Usually, a variety of other entities are used during a model building step such as research questions, qualitative models, theories, assumptions, requirements, and data. If the used model entity belongs to a different simulation study than the generated model, the pattern describes the extension or refinement of an existing, typically already validated model.

"Creating Simulation Model" Pattern: When a simulation model is created from scratch no simulation model is used. However, analogously to the refining pattern, various entities can be used: research questions, qualitative models, theories, assumptions, requirements, data, or other information sources. The output of the "creating simulation model" activity is an initial simulation model that is either entirely new or just has not been linked to another study yet.

"Re-Implementing Simulation Model" Pattern: Sometimes models are not refined or extended but re-implemented using other tools or languages. When re-implementing a simulation model, the provenance graph reflects this as a model-building activity, i.e., a simulation model is used, and another simulation model is generated by the activity. But in contrast to extending or refining, no new conceptual materials (such as a qualitative model, or input data) are used.

"Composing Simulation Model" Pattern: The composition pattern involves two simulation models as input to an activity (used-dependency). These are fused into a composed model, i.e., the output of this activity. Similar to the "refining or creating simulation model" pattern, further entities can be used that deliver context information about why and how to combine the models. Moreover, the used models and the composed model may belong to different simulation studies.

"Calibrating Simulation Model" Pattern: In a calibration experiment, the model parameters are fitted with the help of additional context data. Thus, the input of the "calibrating simulation model" pattern consists of a simulation model as well as data, or alternatively a requirement entity (if the requirement, e.g, expresses properties of the target trajectory and a distance measure). The output of a calibration activity is a simulation experiment containing the experiment specification, a data entity containing the simulation output and the calibration status (success/failure), as well as a modified (fitted) simulation model. Of course, further entities are allowed as input to the calibration, e.g., assumptions, theories or input data, but these are not required. Also, simulation experiments may be used as input to a calibration. This indicates that the new experiment was created by reusing a previous experiment.

"Validating Simulation Model" Pattern: The "validating simulation model pattern" looks in large parts similar to the calibration patterns. Here, also either a data or requirement entity is needed. As in the case of calibration, other entities can be used, e.g., to input data containing the initial configuration, or a reused simulation experiment. Products of the validation activity are a simulation experiment and simulation data indicating the success or failure of the validation. In contrast to calibration, no new simulation model is generated.

"Analyzing Simulation Model" Pattern: An "analyzing simulation model" activity uses a single simulation model as input, produces a simulation experiment and simulation data. In contrast to calibration and validation activities, they do not require specific input entities and their results are not interpreted as success or failure.

"Sensitivity Analysis" Pattern: The patterns for calibrating, validating, and analyzing the simulation model may be refined to match experiments of a particular type. Experiment types

usually cannot be distinguished by a specific pattern structure but rather by the metadata of the involved entities. For example the analyzing simulation model pattern can be refined to a "sensitivity analysis" pattern by checking the information model of the experiment entity for a specific attribute value. This, of course, assumes that the experiment type was made explicit in the experiment entity's metadata.

"areEqual" Pattern: This pattern takes two simulation models. The two models are considered to be equal if their IDs, contained in the information models, are equal. This pattern can be applied to ensure that the used trigger pattern and experiment pattern match the same simulation model entity. It is applied in the case study, see the rule depicted in Figure 5.7.

"isValidated" Pattern: This pattern is used to describe that a simulation model is validated. A simulation model is considered to be validated if a simulation experiment exists that was executed with this model, and the corresponding data entity shows the validation status "successful" in its information model. In the case study, this pattern is applied in the rule depicted in Figure 5.8.

"isBasedOn" Pattern: This pattern relates two simulation models. A simulation model SM' is said to be based on another simulation model SM if a directed path exists in the provenance graph from SM' to SM. In the case study, this pattern is applied in the rule depicted in Figure 5.8.

"haveSameOrigin" Pattern: This pattern is used to express that two simulation models (SM' and SM") have a common predecessor model from which their development started. This is the case if there exists a third simulation model SM to which from both SM' and SM" exists a path, and SM' is not based on SM", and SM" is not based on SM'. This pattern may be applied when generating simulation experiments for analyzing alternative model variants (i.e., implementations in different modeling languages or models capturing different hypotheses about the mechanisms of the modeled system).

Construction of the Rules

The reuse of simulation experiments will be based on a graph transformation system that produces new simulation experiments and associated nodes and edges in the provenance graph based on patterns of the last modeling activity and previously conducted simulation experiments. The graph transformation system is given by a set of production rules R, the so-called reuse rules. A reuse rule is given in the form $(T, E, f) \rightarrow E_{gen}$, where the left-hand side of the rule consists of a trigger pattern T, an experiment pattern E, and a condition function f relating the two to each other and the rest of the provenance graph.

A pattern in RASE can be represented as a set consisting of different "provenance objects", i.e., entity nodes, activity nodes and relationships. A trigger pattern is a provenance graph that is currently considered to contain exactly one activity node and one entity node of type simulation model that was generated by that activity (i.e., a relationship of type wasGeneratedBy exists between the simulation model and the activity). This definition may be extended but currently experiment reuse is assumed to occur only after changes in the simulation model. Defined similarly, an experiment pattern is considered to contain exactly one activity node, however, it must also contain one entity node of type simulation experiment with a wasGeneratedBy relationship to the activity.

Besides trigger patterns and experiment patterns, additional patterns are needed to express conditions on the relationship between trigger pattern (including the most recent simulation model) and experiment pattern (including the experiments to be reused), or to provide constraints for either of these patterns and how they relate to the rest of the simulation study. Consequently, a condition pattern C is an arbitrary provenance pattern that overlaps (i.e., shares some nodes or

relationships) with either the trigger pattern T, experiment pattern E, or both, i.e., $C \cap (T \cup E) \neq \emptyset$. To combine multiple conditions within the same rule, each rule is assigned a condition function f, which is a Boolean function composed of predicates and logical connectives. Each predicate refers to a condition pattern, e.g., one of the predefined patterns in Figure 5.2. There, the predicate $p_1 =$ *isBasedOn(Simulation Model, Simulation Model)* is given by a pattern describing a path from one simulation model to the other. Similarly, $p_2 =$ *isValidated(Simulation Model)* refers to the pattern of an activity, in which some data has been successfully reproduced or a requirement has successfully been checked. A predicate will evaluate to true if matching instances of the condition pattern are found in the provenance graph. An example of condition function f then would be, e.g., to express that, given the predicates p_1 and p_2, that one condition should match while the other should not ($p_1 \wedge \neg p_2$).

The right-hand side of a rule then describes how the provenance graph defined on the left-hand side is transformed. The transformation involves creating an extended provenance graph with a new activity, new entities, and relationships between those. The extension is based on an experiment pattern E', resulting in an extended provenance graph defined as $E_{gen} = T \cup E \cup E'$. E' can be an experiment pattern either of the same kind or different kind as E. This way the new experiment activity can be generated and later be recognized and reused in future model development cycles. Moreover, it makes explicit which nodes of the trigger or experiment pattern will be used within the new activity. Note that thereby rules that modify or delete existing nodes are not allowed—the transformation rules solely extend the provenance graph.

For the framework, a set of rules are predefined that map to the scenarios from Section 5.3 using the patterns from Section 5.4.2. However, there might be domains where these patterns and rules do not apply. Therefore, the framework is designed such that the rule set can be customized by enabling or disabling rules depending on the user's level of expertise. Furthermore, custom rules can be created for specific settings by combining trigger patterns, experiment patterns, and condition patterns as needed. Also, new patterns can be defined.

5.4.3 Rule Matching

The following describes a rule matching algorithm designed to generate a new simulation experiment and transform the provenance graph accordingly. This procedure involves matching the defined patterns within the given provenance graph. Practically, this is accomplished using graph queries. After presenting the rule matching algorithm, two specific queries are introduced. These queries are essential for the algorithm, as they extract all matches from the left-hand side of the rule for reuse and adaptation.

Rule Matching Algorithm

The rule matching procedure is given in Figure 5.3. It is started each time RASE recognizes that a new activity including dependencies and newly generated entities was completed and added to the given provenance graph. Thus, the algorithm takes as input the current provenance graph G, the most recent activity node a, and the reuse rules R defined before. As a first step, it initializes an empty provenance graph N for storing all entities, activities and dependencies that are going to be generated (l. 1). Then, the algorithm iterates through the set of reuse rules R (l. 2). For each rule, first, it tries to match the trigger pattern T at the most recent activity a in G (identified by its *id*, see l. 3).

If a match (i.e., an instance) t of the trigger pattern T is found at the current activity (l. 4), next the procedure checks for suitable simulation experiments to reuse and adapt. All occurrences e of the experiment pattern E for which the condition function f evaluates to true in relation to the matched trigger t are collected in the set M (l. 5). Depending on the rules and the provenance graph, often multiple reusable experiments will be found, for example, various validation experiments where each checks a different behavioral property of the model given by a

Input: a provenance graph G, a set of reuse rules R, the current activity a

```
1  N ← ∅
2  for each (T, E, f) → E_gen in R do
3      t ← find match t of T in G where activity id of t equals id of a
4      if t ≠ NULL then
5          M ← find all matches e of E in G where f(t, e) = true
6          for each e in M do in parallel
7              s ← get experiment specification from e
8              s′ ← adapt s based on t,e
9              d, m ← execute s′
10             E_new ← instantiate E_gen
11             add experiment specification s′ to E_new
12             add result data d to E_new
13             add simulation model m to E_new
14             N ← N ∪ E_new
15 wait for all parallel tasks
16 G ← G + N
```

Figure 5.3: Rule matching at a given activity a. Reprinted from [237].

requirement entity. These experiments can be reused, adapted, and executed in parallel, as the results of experiments conducted with the same trigger model are independent of one another. For the same reason, the rules do not have to be evaluated in a specific order.

For each of the matched experimentation activities $e \in M$ (l. 6), first the old experiment specification s is retrieved from e (l. 7). Second, an adapted version s' is generated taking the context of t and e into account (l. 8). Subsection 5.4.4 describes the adaptions in detail. Then, the adapted experiment specification is executed with the new model, and the simulation data d and possibly a new fitted model m are obtained (l. 9). The generation pattern E_{gen} from the right-hand side of the rule provides the blueprint for creating the new entities, activities, and dependencies. By instantiating this blueprint with actual nodes and edges, a new subgraph E_{new} is created (l. 10). To complete the new provenance subgraph, the simulation results have to be stored in the information model of the new simulation data entity, the generated experiment specification has to be stored in the new simulation experiment entity, and if available, the new model specification has to be stored in the new model entity (ll. 11–13). Finally, E_{new} is temporarily added to the set N (l. 14).

When all rules have been evaluated, the collection of new subgraphs can be added at once to the provenance graph, i.e., $G = G + N$ (l. 16). Before that, however, the algorithm has to wait until all experiment executions are completed (l. 15). This is necessary since a rule might produce a new simulation model and thus would immediately trigger a new round of rule matching. In case the next experiments to be generated depend on the results of the previous experiments, the simulation results should be available before the algorithm proceeds.

The complexity of the matching procedure depends on the number of rules to be evaluated, the size of the provenance graph and the provenance patterns to be matched. To support, e.g., the scenarios described in Section 5.3, only seven rules are required (where the validation scenario accounts for two rules: model extension and model composition). Further, the approach works on aggregated provenance where only the macroscopic steps are viewed, i.e., model building, model calibration, model validation, and model analysis. These high-level provenance graphs and query patterns are easily manageable for state-of-the-art graph databases. Overall, in light of the runtimes required to execute the experiments themselves, which can easily add up to several

hours for a single simulation run, the time required for rule matching is negligible.

Pattern Matching using Graph Queries

A graph query language, such as Cypher [383], can be used to retrieve all subgraphs of the provenance graph that match the defined patterns. A query allows selecting specific types of nodes and relationships, searching for specific sequences, filtering based on certain conditions, and grouping the nodes. Various illustrating examples for the syntax of the Cypher query language were provided in Section 4.3.5. An example for pattern matching the left-hand side of a reuse rule is given by the following queries. They refer to matching the rule $r2$ used in the case studies (reuse of an analysis experiment for cross-validation after the model has been calibrated, see Section 5.6). The first query represents the trigger pattern to match a model calibration:

```
1  MATCH (r)<--(a:ACTIVITY)<--(e:EXPERIMENT)
2  WHERE (r:DATA OR r:REQUIREMENT)
3  WITH r,a,e
4  MATCH (s:SIMULATIONMODEL)<--(a)<--(d:DATA)
5  WITH r,a,e,s,d
6  MATCH (a)<--(outs:SIMULATIONMODEL)
7  WHERE a.id = 'CSM1' AND a.studyName = 'Haack et al. 2015'
8  RETURN s, r, a, e, d, outs
```

It matches an activity that takes data or a requirement and produces an experiment (ll. 1–2). Like any simulation experiment activity, the activity also should take as input a simulation model and produce simulation data (ll. 3–4). Characteristically for calibration, in addition, a calibrated simulation model should be produced (l. 5–6). To match only the latest simulation model of the current simulation study, the result is filtered for the activity *id* (we assume that the experiment generator is notified about any new activity) and the study name (l. 7). The query returns all nodes belonging to this activity (l. 8).

The second query represents the experiment pattern to be matched for reuse:

```
1  MATCH (m:SIMULATIONMODEL)<--(a:ACTIVITY)<--(e:EXPERIMENT)
2  WHERE ((m)<-[*2..]-({id: 'SM1'})) AND NOT m.studyName = 'Haack et al.
     2015' AND m.isValidated = true AND NOT (a)<--(:SIMULATIONMODEL)
3  WITH a,m
4  MATCH (y)<--(a)<--(d:DATA)
5  WHERE d.status = 'undefined' AND NOT (y:SIMULATIONMODEL)
6  RETURN a, m, e, d, collect(y)
```

It matches an experimentation activity (characterized by the path defined in l. 1) that meets the following conditions (l. 2): 1. the model m used is a predecessor to the simulation model matched by the trigger pattern (in the case study, this model has the identifier "SM1"). In other words, SM1 is based on m, and the path between these models is at least 2 edges long; 2. the analysis was part of a different study (i.e., not part of the current study named "Haack et al. 2015"); 3. the model used in the earlier experiment has already been validated. In addition, experiments used for calibration and validation are excluded, i.e., the activity is not allowed to produce a simulation model (see end of l. 2) and the analysis data cannot contain a validation status (ll. 3–5). Thus, only "analyzing simulation model" activities will be part of the results. The query returns all matches including the involved nodes (l. 6). This includes collecting all entities—other than the simulation model—that were used as input. These are stored in the variable y (ll. 4–6).

Note that building these queries can be rather complex. Therefore, the implementation of the patterns (and thus the queries) is hidden from the users. Users typically only need to be concerned with selecting predefined rules for supporting their study, or with constructing new rules from existing patterns. Depending on the rule definition, RASE then can build the necessary queries on demand from smaller building blocks, i.e., the predefined trigger patterns, experiment

Table 5.2: Initial concentrations for variables in the Wnt signaling model by Lee et al. [32]. The table is enhanced with ontology tags from the UniProt knowledge base (UniProtKB) and the Unit Ontology (UO).

Name	Value	Unit
APC (UniProtKB:P70039)	100	nM (UO:0000065)
GSK (UniProtKB:Q91757)	50	nM (UO:0000065)
Axin (UniProtKB:Q9YGY0)	0.02	nM (UO:0000065)

patterns, and condition patterns. Also, the study-specific information (for instance, "SM1" and "Haack et al. 2015") will be inserted on demand. Only when new patterns shall be defined and used in a rule, new Cypher code needs to be implemented.

5.4.4 Adaption of the Experiment Specifications

As seen in the algorithm depicted in Figure 5.3 (l. 8), adapting the old experiment specifications extracted via the experiment patterns is a crucial step. There are two reasons: First, they may not be executable anymore with the current model (version) due to syntactic issues and incompatibilities of formats and tools. Second, they may not correctly reflect the current model's contents and context (e.g., neglect an important parameter in a parameter scan). To evaluate whether changes have to be made to the experiment specification, the contexts of the old and new simulation model need to be taken into account. The term context subsumes all entities that participated in the model building activities and other activities, such as a successful model validation. The entities that are of particular interest for the adaption of simulation experiments are the qualitative model, data and information sources, research questions, as well as the assumptions and theories, which are part of the conceptual model [310]. The adaption component of RASE traces these entities in the provenance graph and checks newer versions (i.e., since the last experiment execution) for relevant information.

In a provenance graph, it is easy to detect whether the context of the model changed. Graph queries allow extracting for which entities a new version was produced or when a new entity was created (via an activity). However, identifying the exact change in context by examining the information models of the involved provenance entities is more challenging. The more structured and substantiated with formal expressions the information models are, the better they can be exploited automatically. Chapter 4 discussed what the contents and structure of the various entities are, and referred to work in the area of conceptual modeling [310], workflows [223], and provenance ontologies [43]. The formalization of metadata was also discussed, with domain-specific languages and ontologies being the proposed approaches.

So depending on what and how information is stored, some changes can easily be detected by simply comparing the information models. For instance, changes in the name of a model or submodel (as given in the qualitative model) or changes in the role of data (e.g., from calibration to validation) can be identified through simple text comparison. However, for other types of information, a simple text comparison is not sufficient. For instance, if a new name is identified in the list of parameters or list of variables of the new qualitative model, the question remains whether this is an entirely new parameter/variable or a renaming of a previously used one. Similarly, if a non-functional requirement requests a "stochastic simulation algorithm", the question is whether this refers to the same algorithm used in the previous experiment with the earlier model or a different algorithm altogether. Also, if the unit of a parameter value changed, without additional information, it is unclear what impact this has on the simulation experiment. In these cases, ontologies can help to disambiguate and provide further the information. Pieces of information can be annotated with tags from various ontologies. These can be domain-specific

```
<listOfSpecies>
  <species boundaryCondition="false" compartment="Cytoplasm" constant="false" id="Dsh_i"
    initialConcentration="100" metaid="COPASI3" name="Dsh_i">
    <annotation>
      <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:dcterms="http://purl.org/dc/terms/" xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#"
        xmlns:bqbiol="http://biomodels.net/biology-qualifiers/" xmlns:bqmodel="http://biomodels.net/model-qualifiers/">
        <rdf:Description rdf:about="#COPASI3">
          <bqbiol:isHomologTo>
            <rdf:Bag>
              <rdf:li rdf:resource="http://identifiers.org/uniprot/P51142"/>
            </rdf:Bag>
          </bqbiol:isHomologTo>
        </rdf:Description>
```

Figure 5.4: Excerpt of an SBML specification of the Lee et al. model ([32]) with references to BioModels and UniProt.

(e.g., the Gene Ontology (GO) [321] and the Universal Protein knowledge base (UniProt) [297]), identify an algorithm (e.g., the Kinetic Simulation Algorithm Ontology (KiSAO) [335]), or refer to methodology of a specific modeling and simulation approach (e.g., the Ontology for Discrete-event MOdeling and simulation (DeMO) [266]), see also the discussion in Section 4.3.1. Table 5.2 shows initial concentrations of the Wnt signaling model by Lee et al. [32]. The table, given in the conceptual model, is enhanced with tags from UniProt and the Unit Ontology [59] to help interpret the specified names and units.

Sometimes, if not enough metadata is available in the entities of the conceptual model, the simulation model specifications need to be analyzed and compared as well, e.g., to detect changes in variable names and initial values. If the model is specified in a structured format, such as SBML, and the format allows for semantic annotations, these can be exploited in a similar manner to the qualitative model or assumptions. Figure 5.4 shows an excerpt of an SBML file created in COPASI. A list of species is specified, which starts with a species named "Dsh_i" and its initial concentration. The species is annotated with the UniProt identifier P51142 using the RDF[60]-based annotation scheme of SBML. To indicate the relationship between the model species and this identifier, an additional annotation with the BioModels qualifier "isHomologTo" is used. The BioModels qualifiers are an ontology for describing the "relation between a model component and the resource used to annotate it"[61].

Structured formats like SBML and CellML also allow for using specialized methods to characterize and interpret changes in model files. An example of such as tool is BiVes, which analyzes the hierarchical structure of XML-based simulation models [406]. It can identify and interpret structural changes, such as the addition of a new entity and their corresponding reactions, the removal of reactions, and other modifications. Additionally, methods for unambiguously visualizing model differences may be convenient in cases where the differences cannot be correctly interpreted automatically, and the user is needed as a tie-break [423].

Otherwise, if no structured format like SBML is used, established version control systems such as Git can be utilized to track syntactical changes in simulation model specifications [28] as well as diff tools for highlighting these changes. These methods can generally be applied to any artifact [28]. However, to be truly useful for automatic experiment adaption, the semantics of the respective languages have to be computationally accessible, so that expressions and their parts can be interpreted automatically and their implications for the simulation experiment can be derived. For instance, when a difference in a requirement specification in MITL is recognized, it is essential to identify the type of change: e.g., whether a new threshold was defined for a species or whether some property was extended to a larger time interval. In the former case, if

[59] `http://obofoundry.org/ontology/uo`, last accessed 19 July, 2024.
[60] Resource Description Framework `https://www.w3.org/RDF/`, last accessed 19 July, 2024
[61] `https://github.com/combine-org/combine-specifications/blob/main/specifications/qualifiers-1.1.md#model-qualifiers`, last accessed 19 July, 2024.

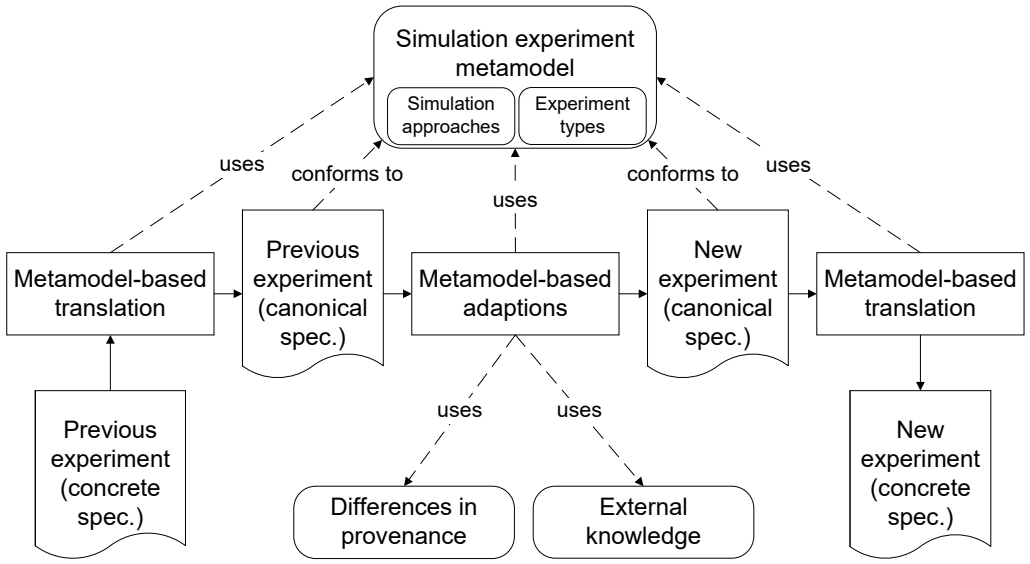


Figure 5.5: Adapting and generating simulation experiments based on metamodels. The pipeline and the metamodels were defined in [17]. Figure adapted from [237].

the experiment is a statistical model checking, simply the formula to be checked needs to be replaced. In the latter case, the observation times for the simulation have to be updated as well.

After the exact changes in the context of the simulation model have been identified, the next step is making the necessary adaptions to the experiment specification. But before any adaptions can be made, all parts of the experiment specifications have to be made easily accessible. The whole pipeline for experiment adaption is shown in Figure 5.5. It is based on the model-driven approach presented in Chapter 3. The previous (concrete) experiment specifications are translated into an intermediate representation—their *canonical form*—based on experiment metamodels. The metamodels define the vocabularies for various simulation approaches (e.g., discrete-event simulation) as well as experiment-type-specific vocabulary (e.g., sensitivity analysis or statistical model checking). Thus, in the canonical form, each part of the experiment specification can be assigned a quasi-standardized identifier, e.g., *parameterValue* for the value assigned to a parameter and *parallelThreads* for the number of parallel threads with which to execute the experiment. Using these metamodel-based identifiers, adaptions can be defined just once for a given modeling and simulation approach or experiment type rather than the variety of different tools with concrete specification languages. An adaption rule defines, depending on the various recognized differences in the provenance (preconditions), how exactly to modify the canonical experiment specification (postconditions). Here additional, external knowledge may be consulted to decide on the exact modifications. After all applicable adaption rules have been executed, the simulation experiment can be translated back from the canonical into a concrete form using the metamodel and transformation rules to the desired backend. Which backend (and thus experiment language) is used, depends on the models and experiments at hand. If experiments are reused within the same simulation study, typically the same language and backend can be used as with the original experiment. In contrast, when an experiment is reused across simulation studies, it is likely that these studies use different tools, and thus the new simulation experiment will be generated in a different language.

Currently, RASE implements several adaptions (as listed in Table 5.3). For each rule, the changes in the provenance entities are specified, along with which properties of the experiment specification need to be adapted and how. Some of the rules are relevant for the following case studies. Others (see the last 2 rows of the table) are included as a discussion of possible adaptions that might be needed during a simulation study. Note that these are just examples and further adaptions can be defined in future extensions of RASE.

Table 5.3: Automatic adaptions of the experiment specifications. The left column describes how the context information given by the entities in the provenance graph changed. Based on these changes the adaptions are evoked. The right column describes the effect these adaptions have on the experiment specification using the vocabulary of the metamodels from Section 3.5.

Change in the provenance entities	Adaption in the experiment specification
The file location of the *simulation model* changed from a local folder to a public repository.	The property *modelFile* is updated accordingly. The *folder* and *fileName* are removed, and instead the *reference* to the repository is included.
The parameter list of the *qualitative model* contains a change in both the name and the value of a parameter. These changes may refer to rate constants or initial concentrations.	In the experiment, the fields *parameterName* and *parameterValue* are adapted for the configuration of the respective parameter.
In the *qualitative model*, the list of output variables was extended with another species.	The *observables* are adapted with a new *observationExpression*. Depending on what information is given in the qualitative model, an *observationAlias* may also be added.
A new *assumption* about the distribution of a parameter was included.	Depending on the experiment type (e.g., if a parameter scan or parameter estimation is generated), the *factorDistribution* as well as the *factorDistributionParameters* are updated for the respective parameter.
There is a modified *behavioral requirement* in which the temporal logic formula has changed to contain an additional variable that must remain within certain bounds over a specified time interval.	If the experiment is a statistical model checking, the temporal logic formula (*propertyExpression*) needs to be replaced. In addition, the list of *observables* needs to be extended with a new *observationExpression*, and the *observationTime* needs to be updated to include the *observationRange* given by the formula.
A *non-functional requirement* was added that indicates the availability of computational resources, i.e., the number of CPU cores.	If the simulation model is stochastic, the experiment specification is updated to include the *parallelThreads* property, which specifies the number of parallel threads to be used during experiment execution.
A new *methodology* is identified in the literature, e.g., for steady-state estimation.	The *methodName* and *methodParameters* of the experiment need to be adapted.
An *assumption* now exists about the behavior of the simulation model, such as assumed linearity of the response surface.	Based on this assumption, more suitable methods for conducting the experiment can be selected. For instance, in the case of sensitivity analysis, a suitable *samplingStrategy* and *sensitivityIndex* can be selected (if knowledge about the different methods has been made explicit, as shown in Appendix B).

5.5 Implementation

The implementation of RASE is openly available in a Git repository[62]. All software components were implemented using Java 8.

The API allows users to connect the framework to any application that produces provenance. This can be done by creating different kinds of provenance nodes and relationships, and sending them to the Prov Model Controller via Provenance Commits (see Figure 5.6). Currently, the high-level provenance model introduced in Section 4.3.1 is assumed, which is implemented in a separate module, and provides the classes Assumption, Data, Experiment, etc. as entity types. Note that the provenance model shown in Figure 5.6 merely reflects the current state of the implementation and may be extended in future work to accommodate new use cases. The recorded provenance is stored in a Neo4j graph database [382]. Accordingly, the queries for the

[62] `https://git.informatik.uni-rostock.de/mosi/exp-generation`, last accessed 19 July, 2024.

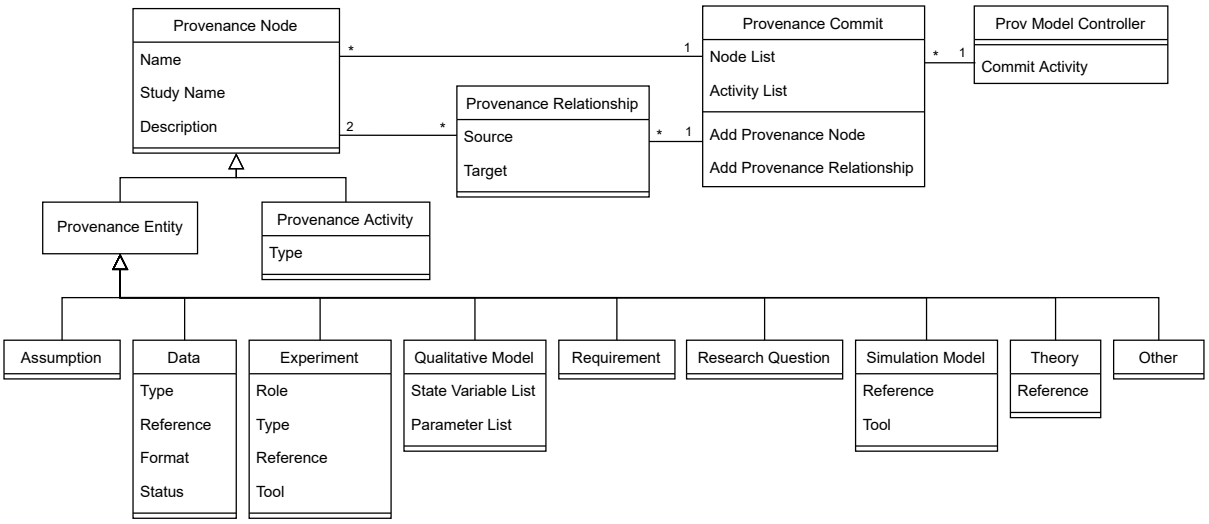


Figure 5.6: API for capturing provenance. Various kinds of provenance nodes and provenance relationships between two nodes can be created and grouped via provenance commits. These can be sent to the graph database via a Prov Model Controller, which may trigger some reuse rules to fire. Provenance nodes can be either activities or entities and have to be filled with meta-data. For instance, the entity type Experiment provides the fields Role (e.g., for calibrating or validating the simulation model), Type (e.g., sensitivity analysis or steady-state analysis), Reference (i.e., the path to the experiment specification), and Tool (i.e., the tool for running the experiment). Reprinted from [237].

various provenance patterns are expressed in the Cypher query language [383].

Within RASE, a previously developed MDE-based toolchain [17] for adapting, generating, and executing simulation experiment specifications is integrated. This allows RASE to handle a variety of specification formats and simulation backends. As intermediate representation for the experiment adaption, it uses JSON[63] documents based on metamodels defined in JSON Schema [273]. For the automatic execution of simulation experiments, it already provides a number of bindings to backends (previously described in Section 3.5), e.g., for SESSL [36] and ML-Rules [40], which are used in the Wnt signaling case study. For the migration case study, an additional binding was implemented for simulation experiments conducted with Julia [70] and R [72].

Note that this chapter focuses on fully automated cases, i.e., everything from recording provenance to executing the generated experiments is done automatically. However, often user inputs are required to correctly adapt the experiment specifications. For this purpose, the experiment generator includes a graphical user interface (GUI) based on JavaFX for presenting the generated experiment specifications to the user and to allow them to make manual changes [17]. The graphical support feature can be enabled via the API. Furthermore, the API allows users to customize the set of provenance patterns and reuse rules via abstract classes, analogously to the implementation of the case studies, to receive the necessary level of support. For users with little programming experience, the implementation of new rules can still be difficult. Therefore, a crucial part of future work will be concerned with evolving the API's usability and documentation.

[63] ECMA-404 The JSON Data Interchange Standard, `https://www.json.org/`, last accessed 19 July, 2024.

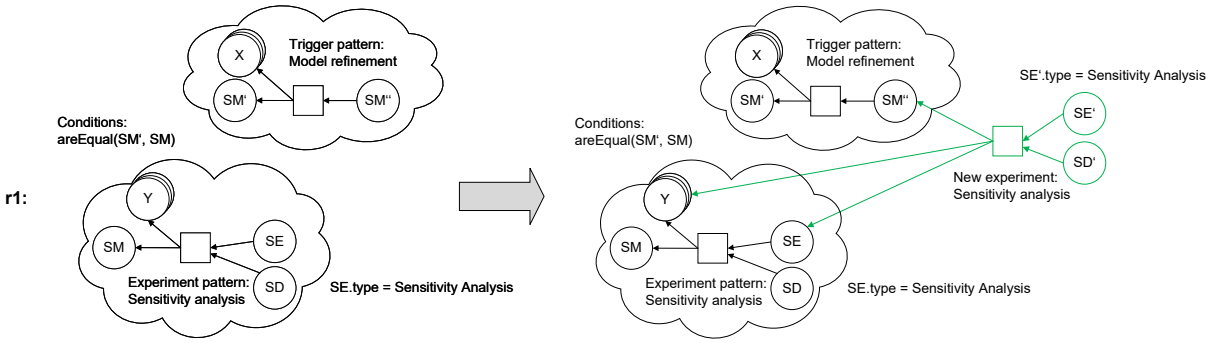


Figure 5.7: The exemplary reuse rule $r1$ describes the repetition of a sensitivity analysis. The left-hand side specifies model refinement as a trigger pattern T, sensitivity analysis as a pattern of a previous simulation experiment E, and an additional condition pattern C to be evaluated to be evaluated as part of the condition function f. The right-hand side (E_{gen}) extends the provenance graph by a new sensitivity analysis experiment (shown in green). Reprinted from [237].

5.6 Case Studies

To demonstrate the usefulness of the approach, two different simulation studies will be conducted. The first case study will show a repeated sensitivity analysis of a model of migration routes. The second case study will reuse an experiment across simulation studies for cross-validation. Afterward, the advantages of the approach in the fields of sociology and cell biology are discussed. To reproduce the case studies, please refer to the accompanying Zenodo publication[64]. Since the case studies can only show a selected set of rules and patterns at work, an additional, abstract simulation study is provided as part of the source code to illustrate further reuse cases based on a predator-prey model.

5.6.1 Configuration of the Rule Set

In RASE, so far, several rules are predefined to cover the scenarios described in Section 5.3. In the case studies, two of these rules are enabled by the users. Figure 5.7 shows the first rule ($r1$), which describes the generation of a sensitivity analysis. It takes a model refinement step as a trigger (T) and the sensitivity analysis pattern for finding previous experiments (experiment pattern E). As a condition (C) for searching in the provenance graph, the simulation model used in the refinement has to be the same model that was used in the sensitivity analysis. From this information, a new activity is generated that takes the latest simulation model (from the trigger pattern), as well as other inputs from the previous experiment activity (accumulated in the multi-entity Y). To denote that an experiment was reused during this step, also the old simulation experiment SE is taken as input. The output of the new activity is a simulation experiment entity with type sensitivity analysis. Moreover, a simulation data entity is generated. The right-hand side of the rule (E_{gen}) describes exactly how this extension relates to the current state of the provenance graph (given by T and E).

The second rule ($r2$), shown in Figure 5.8, presents the cross-validation scenario. Here, model calibration is used as a trigger activity. Calibration is allowed as a trigger (T) since it produces a new (calibrated) simulation model. If a model was calibrated, suitable experiments to reuse are identified using the analysis pattern (E). However, further condition patterns (C) have to apply, i.e., the simulation model used in the trigger pattern has to be based on the simulation model used in the analysis pattern, they have to belong to different studies, and the model used in the analysis has to be validated. Joined by the logical connective $\wedge$, these patterns

[64] `https://doi.org/10.5281/zenodo.6792024`, last accessed 19 July, 2024.

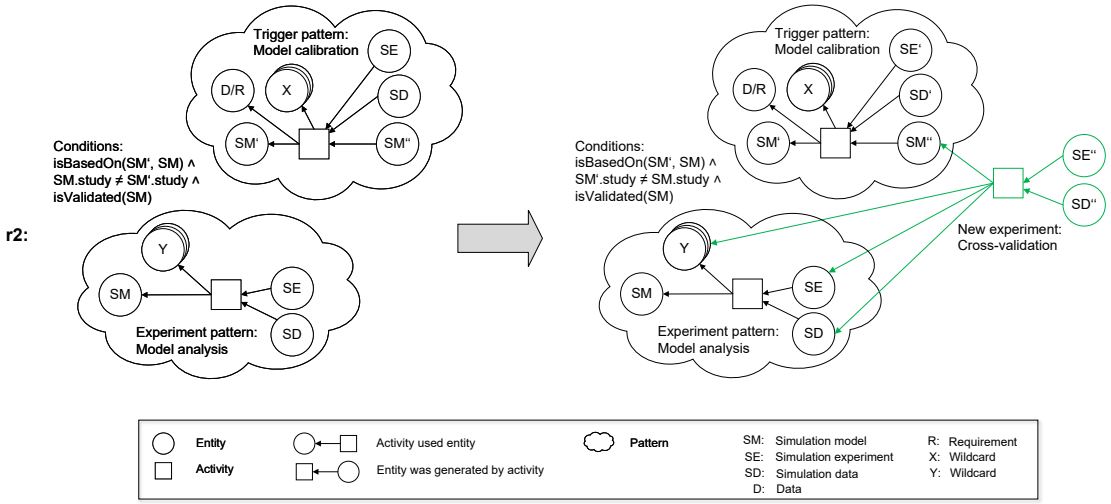


Figure 5.8: The exemplary reuse rule $r2$ shows the cross-validation scenario. The left-hand side specifies model calibration as a trigger pattern T, model analysis as a pattern of a previous simulation experiment E, and additional condition patterns C that, joined by logical connectives, form the condition function f. The right-hand side (E_{gen}) extends the provenance graph by a new cross-validation experiment (shown in green). Reprinted from [237].

form the condition function f for the rule. On the right-hand side (E_{gen}), it is described how a provenance graph that matches the left-hand side needs to be extended. The newly generated activity, shown in green, corresponds to the validation pattern. It is linked to the simulation model from the triggering activity, as well as the inputs, simulation experiment and simulation data from the analysis experiment. In contrast to the generation part in $r1$, here the output data of the previous experiment is taken as input to allow for comparison of the two models, which is crucial for cross-validation.

5.6.2 Repeated Analysis of a Migration Model

The first case study refers to an agent-based model of asylum migration to Europe [69] focusing on the formation of migration routes in response to spread of information. It aims to connect individual decisions (information transfer) on the micro level to processes observed at the macro level (variability and optimality of migration routes). Models are successively refined and analyzed[65] in response to developing theoretical lines of inquiry and the introduction of different data sources, which presents various opportunities for the framework to automatically reuse and generate simulation experiments. The provenance of the study was described in [73] and [301]. Here, only the parts that focus on the modeling and analysis activities are shown (Figure 5.9). Everything concerning psychological experiments and data processing is omitted. Furthermore, the provenance graph was slightly modified to make the experiment specifications explicit (SE1-SE4).

The simulation study begins with the creation of a model SM1, which already includes the agents' knowledge about the environment, social networks, and information exchange with discrete-time behavior, and model SM2, which alters the simulated world from a grid-based layout to a more general and less densely connected graph topography [68].

During the model building steps BSM2' and BSM3, different versions of the originally time-stepped model (SM2) are created to obtain more realistic time courses. Model SM3 then presents

[65] SM1–SM2: `https://github.com/mhinsch/RoutesRumours`, SM3–SM5: `https://github.com/mhinsch/rgct_data`, last accessed 19 July, 2024.

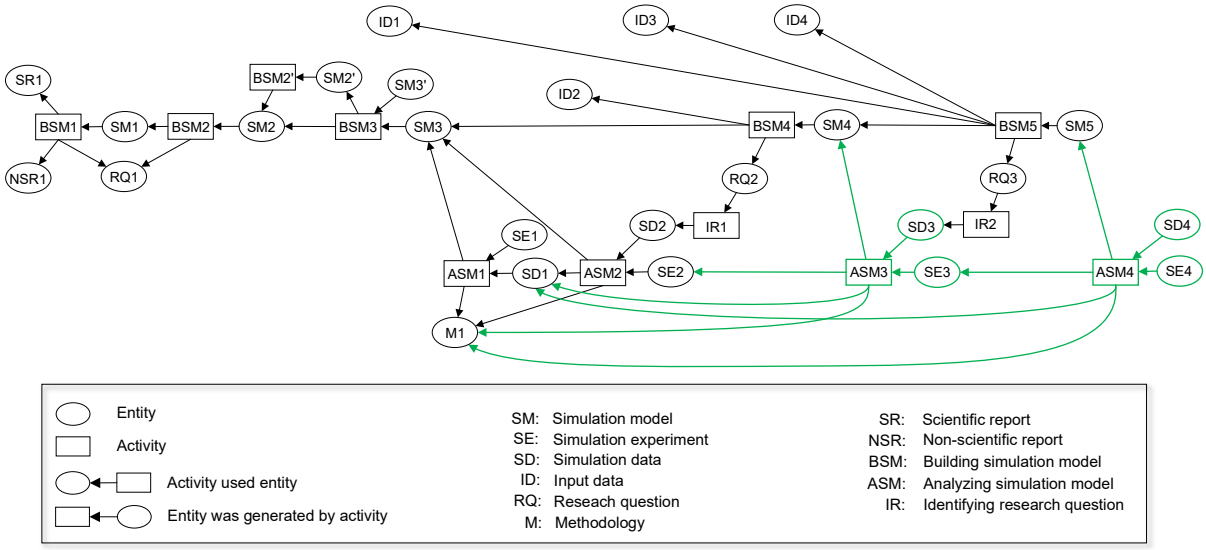


Figure 5.9: Provenance graph of the migration study published in [73] and [301]. Activities, entities, and dependencies shown in green are generated automatically by the approach, and refer to reused sensitivity analyses. Adapted from [237].

a continuous-time version of SM2, implemented in Julia [70]. With this continuous-time model SM3, now first experiments are conducted.

In ASM1, a parameter scan is used to reduce the 17 parameters to the 6 most influential factors. Then, in ASM2 a sensitivity analysis is conducted on the selected parameters. The analysis uses a Latin Hypercube sample to build Gaussian process emulators and carry out the uncertainty and sensitivity analysis. In the original publication, the analysis was conducted with the (GUI-based) tool GEM-SA [202]. For this case study, the same experiment was prepared as R scripts to have the experiment specifications accessible to the approach[66].

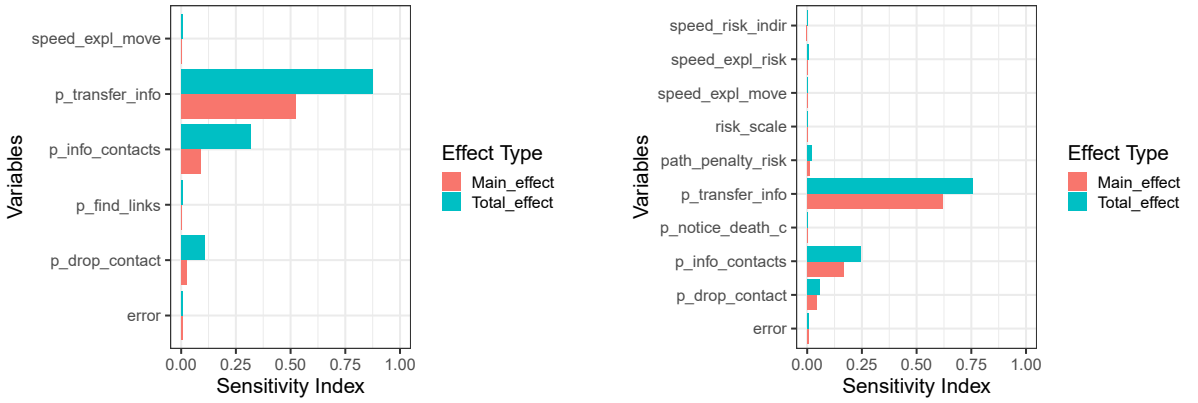
Following the experiment, the next model version is developed to include additional empirical data on information exchange and risk behavior, i.e., during the model refinement step BSM4, the newly identified research question RQ2 and input data ID2 derived from psychological experiments are used. This model refinement now triggers the rule matching of the approach. Rule $r1$ (Figure 5.7) can be applied at activity BSM4 and the following match is found:

$$match_1 = \{SM' \leftarrow SM3, X \leftarrow \{RQ2, ID2\}, SM'' \leftarrow SM4, \\ SM \leftarrow SM3, Y \leftarrow \{M1, SD1\}, SE \leftarrow SE2, SD \leftarrow SD2\},$$

where the left-hand side of the arrows represents the variable names given in the rules, and the right-hand side represents the names of the matched entities from the provenance graph in Figure 5.9.

Executing the rule generates a new activity ASM3, and used-dependencies are drawn to the new simulation model SM4, the previous experiment SE2, and other input entities involved in ASM2, i.e., the analysis methodology presented by Kennedy and O'Hagan (M1 [136]) and the simulation data SD1. As output, according to the "sensitivity analysis" pattern, an experiment entity SE3, and a simulation data entity SD3 are generated. When generating the new experiment specification for entity SE3, the model name has to be adapted from "SM3" to "SM4". In addition, the list of parameters is updated to include also the factors concerning risk behavior. Running the experiment produces main and total order sensitivity information, which is added to the information model of SD3. Figure 5.10 shows the sensitivity data for SM3 (left) and SM4 (right) for the output mean_freq_plan, which captures the proportion of time for which

[66] `https://github.com/jasonhilton/screen_run`, last accessed 19 July, 2024.



(a) Main and total effects for M3

(b) Main and total effects for M4

Figure 5.10: Sensitivity indices calculated with the original experiment and model, and the automatically reused experiment and model. Reprinted from [237].

agents are following their plan for transiting the space. In both model versions, the parameter p_transfer_info was identified as the key driver for planning behavior, indicating that information transfer was crucial for plan choice. However, it is noticeable that the addition of risk behavior in SM4 does not have any substantial influence on this particular output.

Following the experiment results, the simulation model is refined further. New data entities and research questions are included in the next model refinement step, which produces the model version SM5. Again, triggered by the refinement activity, the approach automatically repeats the previous sensitivity analysis experiment. This time the activity BSM5 is taken as the anchor point for matching the rule $r1$. The activity ASM4 is automatically generated including the new entities SE4 and SD4, and new used-connections to the inputs M1, SD1, and the previous experiment SE3.

5.6.3 Cross-Validation of two Models of the Wnt/β-Catenin Signaling Pathway

The Wnt/β-Catenin signaling pathway is a central pathway in the development and homeostasis of cells [29]. Degenerated forms of this pathway are involved in a number of cancers and neurological disorders [30]. The study by Haack et al. (2015) analyses the regulation of Wnt signaling in the initial cell fate commitment phase of neuronal progenitor cells [31]. In-vitro experiments indicated that raft- and redox-dependent signaling events play a crucial role in the regulation of Wnt signaling during this phase. A previous simulation model of the Wnt signaling pathway by Lee et al. (2003) [32] does not contain the corresponding model entities to accurately represent these regulatory mechanisms. Therefore the Lee model was extended with a membrane model component as well as additional intracellular model entities and mechanisms. The extended model was calibrated against new in-vitro data and subsequently cross-validated with simulation data from the Lee study to ensure that the basic model behavior was not changed due to the extension and (re-)calibration of the model.

Provenance for both simulation studies was documented by [43]. Figure 5.11 (left) depicts the entire provenance graph of the Lee study. For the Haack et al. study, only the initial phase with the first few activities (model building BSM1 and calibration CSM1) until the first automatic experiment generation is shown. The connection between the models is shown by a used relationship between SM2 of the Lee model and BSM1 of the Haack model. Following the model extension, a calibration experiment is conducted. This is recognized as a trigger by the approach[67].

[67]The resources for the Wnt case study are provided with the main repository: `https://git.informatik.`

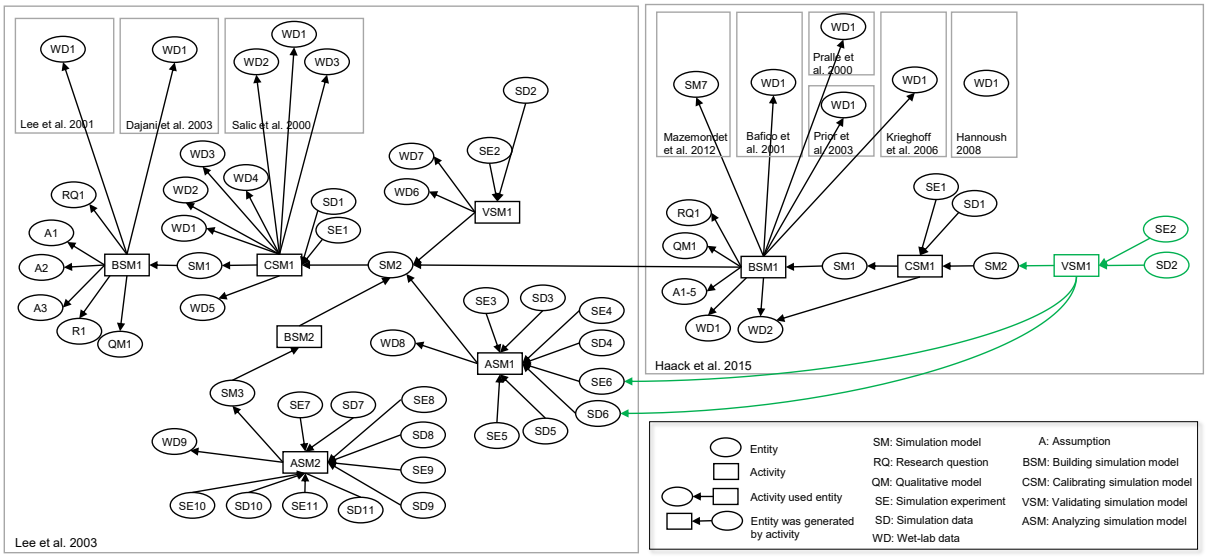


Figure 5.11: Provenance graph showing the simulation study by Lee et al. (2003) and the initial phase of the Haack et al. (2015) study. Both studies are related via the model building activity BSM1. Following the calibration activity CSM1, cross-validation is triggered for the new model SM2. Thus, a new validation activity VSM1 is generated which reuses the simulation experiment SE6 and the simulation data SD6 from the Lee study and produces an adapted simulation experiment SE3 and the simulation data SD3. The automatically generated entities, activities, and relationships are shown in green. Adapted from [237].

In particular, the cross-validation rule $r2$ (Figure 5.8) can be applied at CSM1. Note that to save space, in Figure 5.11 activities of the same type are displayed in aggregated form (e.g., ASM1 of the Lee study aggregates four experiment activities). Thus, this yields four different matches of the rule. E.g., $match_1$ matches SE5, SD5, and input wet-lab data WD8 as suitable candidates for reuse, and $match_2$ matches SE6 and SD6:

$$match_1 = \{SM' \leftarrow SM1, D \leftarrow WD2, X \leftarrow \emptyset, SE' \leftarrow SE1, SD' \leftarrow SD1, SM'' \leftarrow SM2, \\ SM \leftarrow SM2, Y \leftarrow \{WD8\}, SE \leftarrow SE5, SD \leftarrow SD5\}$$
$$match_2 = \{SM' \leftarrow SM1, D \leftarrow WD2, X \leftarrow \emptyset, SE' \leftarrow SE1, SD' \leftarrow SD1, SM'' \leftarrow SM2, \\ SM \leftarrow SM2, Y \leftarrow \emptyset, SE \leftarrow SE6, SD \leftarrow SD6\}.$$

Figure 5.11 exemplarily shows the generated validation activity based on SE6 and SD6 of the Lee et al. study and the new model SM2 of the Haack et al. study. The simulation experiment SE6 was specified in SED-ML and the corresponding model SM2 of the Lee et al. study in SBML [283]. The files[68] are available on the BioModels database [42]. As discussed in Section 4.3, not only the artifacts but also entire provenance graphs of related simulation studies may be available from model databases like BioModels in the future.

As the Wnt model by Haack et al. was specified using the rule-based modeling language ML-Rules [40], and the experiments are conducted using the experiment specification language SESSL [36], during the adaption and generation step, the experiment specification has to be translated from SED-ML. During the translation, to receive an executable experiment specification, it is checked whether the observed species are still known under the same name in the extended model. Here, the qualitative models (both denoted QM1), based on which the simulation models

`uni-rostock.de/mosi/exp-generation`, last accessed 19 July, 2024.

[68] `https://www.ebi.ac.uk/biomodels/BIOMD0000000658`, last accessed 19 July, 2024.

Table 5.4: UniProt tags (prefixed by "UniProtKB") are used to identify the species involved in the Lee model, and to map them to species in the Haack model. Adapted from [237].

Lee Model	Haack Model	Ontology Tag
W	Wnt	UniProtKB:P31285 (WNT3A_XENLA)
Axin	Axin	UniProtKB:Q9YGY0 (AXIN1_XENLA)
Beta-catenin	Bcat	UniProtKB:P26233 (CTNB1_XENLA)
Dsh	Dvl	UniProtKB:P51142 (DVL2_XENLA)

were built, come into play. The qualitative models contain a list of species, each annotated with proteome identifiers from the Universal Protein knowledge base (i.e., the ontology UniProt) [297]. Table 5.4 shows a mapping of selected model species from the Lee and Haack models. In the case of Beta-catenin, which is the variable of interest in this experiment, a translation is required.

But also other aspects of the experiments or models might have to be considered when reusing an experiment from a different simulation study. For instance, in this case study, the models are subject to different time scales. This is because the parameters of the models were fitted against experimental data from Xenopus egg extracts (Lee et al.) and human neural progenitor cells (Haack et al.), two cell-biological systems with different time scales. Since differentiation happens faster in human neural progenitor cells, the reaction rates of the Haack et al. model need to be scaled to compensate for this difference [31]. Thus, the experiment specification needs to apply a constant scaling factor to all parameter values of the model. To find the right scaling factor (0.28) automatically, a simulation-based optimization needs to be performed, fitting the output of the Haack et al. model to the data from the Lee et al. model.

After these adaptions, the new experiment SE3 is generated and can finally be executed. Figure 5.12 shows the cross-validation results. It compares the trajectories of the key protein β-catenin, an indicator of the pathway's activity, produced by the Lee and the Haack model (with and without adapted time scale) when stimulated with a transient stimulus. After applying the correct transformation factor, that corrects for the varying time scales, the β-catenin curves show the same maximum at the same time. This means that the extended model developed by Haack et al. still corresponds to the core dynamics of the Wnt signaling pathway modeled by Lee et al.

An alternative to the straightforward comparison of time series would be a cross-validation using statistical model checking. In this case, the simulation experiment conducted by Lee et al. would need to be automatically rewritten to adopt a different experiment type (statistical model checking) and the required output data would need to be expressed, e.g., in terms of the minimum and maximum values of β-catenin in a specific time range. Listing C.1 shows generated SESSL code for such an experiment.

5.6.4 Results

In both case studies, RASE could successfully demonstrate how provenance information can be exploited in conducting simulation studies in a more systematic and effective manner. Changes (or lack of changes) in the sensitivity of simulation outputs to parameters following substantive alterations to the model provide important information about the mechanisms at work in a simulated system. In the case of the migration model above, changes in the decision-making process of agents did not hugely affect the output in question. The ability to automatically identify the experiment needed for sensitivity analysis (and subsequently to trigger this experiment following a change in the model) significantly shortens the modeling cycle and reduces the burden on the modeler. This means that the focus can be on analyzing and interpreting simulation results and considering additional modeling steps.

In addition to shortening the modeling cycle, automated experiment specification is also

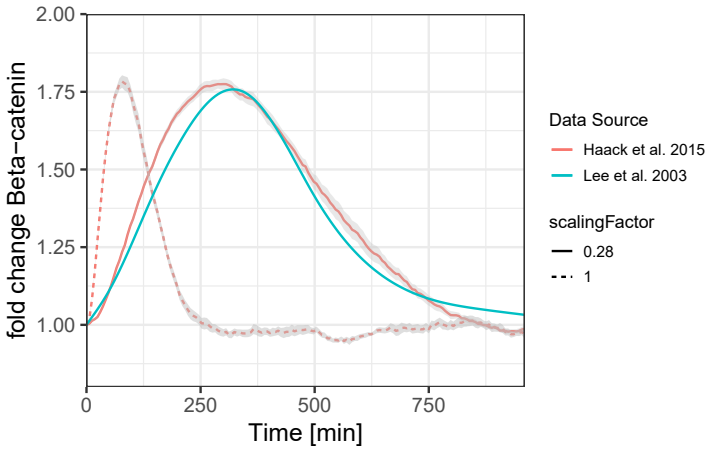


Figure 5.12: Results of the cross-validation experiment. Beta-catenin values are compared between the Lee model and the Haack model before and after applying the scaling factor. Reprinted from [237].

valuable to increase the models' validity and reusability. For instance, in the Haack study, an existing model was extended by further model components that significantly altered the structure of the model. However, despite the changes, the model should still be able to reproduce basic dynamics of the pathway that were either obtained in wet-lab experiments or by simulations of a previous model version. The ability to automatically identify and reuse all experiments that are necessary to establish the validity of a model allows for easier and more rigorous validation of extended models.

5.7 Discussion

The approach implemented in RASE relies on two requirements: 1) on documenting simulation studies as graphs following the provenance standard PROV-DM, 2) on executable simulation experiments, simulation models, and input data as part of or referenced in the graphs, 3) additional explicit context information about the simulation study to make the adaptions.

Referring to requirement 2, the last decades have seen increasing efforts of documenting simulation studies and making the diverse artifacts, including the simulation experiments, available to ensure the reproducibility and credibility of study results. These can now be exploited automatically. The systems biology community, e.g., made substantial progress in standardization [424, 425]. This includes languages for specifying simulation models [283] and simulation experiments [193]. Moreover, standardization extends to the way the community packages and shares simulation models and other artifacts: a COMBINE archive [253] is a single file that can comprise separate model and experiment files as well as data, metadata, and images. Also in the agent-based modeling community (executable) artifacts are shared, e.g., via the model database OpenABM [256] and electronic lab notebooks [326].

Regarding requirement 3, knowledge about simulation approaches, application domains, and methodologies needs to be encoded. Here, in particular, the use of ontologies was discussed. The systems biology community is leading in the development of ontologies, having developed various ontologies for the annotation of biological models, experiments, and data (e.g., [397, 297]). However, other domains of simulation, such as logistics [426] or physics [267] also have developed relevant ontologies. Despite these developments, what is often missing is explicit information about the relationships between the concepts of different ontologies. Addressing this gap, and thus further improving the automatic reuse and adaption of simulation experiments, requires significant community efforts.

Regarding requirement 1, the provenance standard PROV-DM comes with plenty of benefits, such as improved visualization and analysis of the simulation study, but is currently not widely used in simulation. However, provenance may also come in a different form. For instance, documentations based on reporting guidelines, such as TRACE [236] and STRESS [246], provide crucial information about a simulation study in the form of verbal narratives. They describe the inputs, outputs, and versions of a model as well as the analyses conducted. Often they refer to platforms such as GitHub, where the described artifacts are made available. To apply the pattern-based approach to studies that use non-standardized provenance, two ways exist.

The first way would be to transform the provenance into the PROV-DM standard. Here, recent approaches for automatically capturing provenance may be applied as a pre-processing step to this framework. These include, e.g., the abstract syntax tree analysis and execution trace analysis of scripts [362], the interpretation of electronic lab notebooks using ontologies [427], and the extraction of facts from scientific articles [428].

The second way would be to recognize patterns in these documentations directly. In TRACE, e.g., it is already annotated whether an experiment was run for calibration or validation, which would facilitate the recognition. COMBINE archives, as another example, could be interpreted as a single experimentation activity that can be reused to generate a new experiment, i.e., a new COMBINE archive. Increasing efforts should be directed towards handling these other forms of documenting simulation studies—outside of the provenance standard. Recently, various provenance tools have been developed, e.g., for provenance management in data science notebooks [429] or the visualization of provenance [430]. Thus, it can be anticipated that in the future also helper tools for provenance transformation and provenance pattern recognition will become available, which will make provenance pattern-based support even more feasible. However, there are domains that use closed-source, all-in-one tooling, for which integration with this framework would still be difficult.

Provenance of simulation studies, and thus the provenance patterns used, were so far centered around the simulation models and simulation experiments. What happened before or in between the model building steps and after the simulation experiments was not considered. For instance, the data sets that were used or generated by the model building or experimenting activities were treated as monolithic entities without an own generation process. In some studies, however, how data was collected and processed or how hypotheses were formed is an integral part of the research, and thus the modeling and simulation life cycle. This is especially the case with data-driven or hypothesis-driven research, as in the social sciences.

In these studies, one can, for example, distinguish the processes related to primary data (i.e., data collected specifically for the current simulation study through interviews, real-world experiments, etc.) and secondary data (i.e., data collected not by the investigator of the current study and for a different purpose, such as data collected by the government or organizations) [301]. The patterns associated with primary and secondary data collection are depicted in Figure 5.13. For primary data collection, the design and implementation of the data collection process is in the focus. Using secondary data, on the other hand, requires assessing and cleaning the data with respect to biases and uncertainty. For the specification of the respective patterns, the underlying provenance model needs to be extended by further entity types. The required entities for primary data collection include methodology literature, data collection procedure, participant information, preregistration documents, ethical approval documents, primary data, findings, and analysis specification. Entities related to secondary data are the assessment framework, metadata, and cleaned data.

With these and other additional activity patterns explicitly specified, it would be worth investigating how a (semi-)automatic support for interlacing simulation experiments with real life experiments and data collection procedures can look like. Especially the reuse of different kinds of data entities (e.g., field data, wet-lab data or survey data) as well as metadata and literature about those seem promising in this endeavor. Here, an earlier simulation experiment

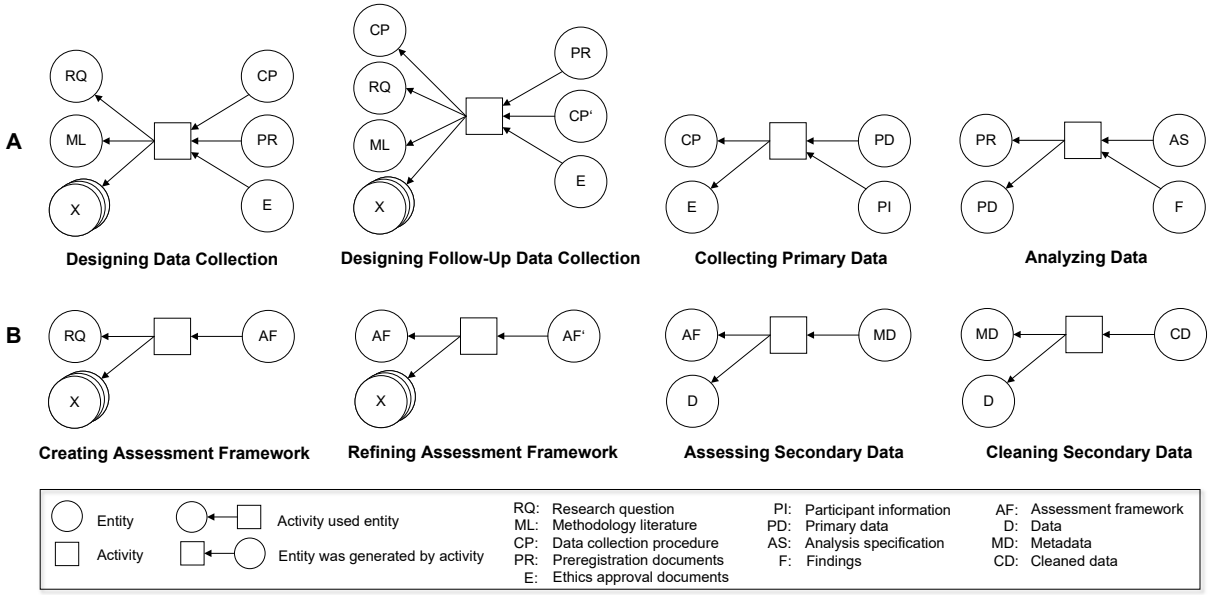


Figure 5.13: Provenance patterns for primary data collection (subfigure A) and secondary data collection (subfigure B). The prime symbol (') is used to denote versions of an entity. Adapted from [301].

may not even be required, but the experiments may be generated from scratch solely based on the data and context given. The following section presents first ideas and challenges concerning the experiment generation from scratch.

Another caveat to consider in the rule-based approach is the cost of excessive automatic reuse. Reasons may be too fine-granular updates in the provenance (e.g., every time the model file is saved) or reuse of older experiments (beyond the most recent versions). Consequently, the rules have to be carefully designed to lend enough support but so that the computational load remains manageable. In addition, mechanisms can be installed that do not reuse and re-run simulation experiments every time a rule matches, but instead prioritize and select a few, most important simulation experiments. The experiments may, e.g., be prioritized based on the expectation of how different the result of the modified simulation experiment will be in comparison to the original one. This expectation may be explicitly provided by the modeler, or based on a similiarity/difference score between the newly generated code and the previous one [431].

5.8 Generating Simulation Experiments from Scratch

Thus far, RASE reused previous simulation experiments, which already gave direction towards the actions to follow. A different approach would be to start from a clean slate, with fewer constraints regarding what to do next, and viewing the generated simulation experiments as possibilities rather than necessities.

However, when generating simulation experiments "from scratch" a myriad of challenges exist that can only be remedied if further knowledge is made explicit about the context of this particular simulation study, about the application domain, and about modeling and simulation in general and the related methodologies. Chapter 4 of this thesis proposed some representations of this diverse knowledge. In RASE, information about the artifacts and activities of the simulation study was expected as a provenance graph based on a broad definition of conceptual model and (partly) formalized metadata. Going one step further, one may also relax the constraints regarding the form in which knowledge is provided, and instead extract relevant knowledge from

unstructured or semi-structured texts. There, existing and constantly improving pre-trained large language models may be facilitated [358], e.g., to derive the structure of the qualitative model, or to identify behavioral requirements.

To support the generation of experiments from scratch, RASE would therefore need to be extended by 1) another type of inference rule that does not expect a previous experiment, 2) access to additional sources of knowledge about modeling and simulation studies, e.g., goal hierarchies to identify when in the simulation study which types of experiments are necessary, and 3) methods for knowledge extraction and pattern recognition in unstructured data.

Some of the following challenges have also been addressed in the case of reusing a simulation experiment, however, when generating simulation experiments from scratch they become even more important.

5.8.1 Central Challenges

What type of simulation experiment to execute? What type of simulation experiment to generate depends on the provided context information. If parameters are given in the conceptual model with a value range, a sensitivity analysis based on these parameters can be executed. If behavioral requirements are defined in terms of data, the requirements can be used for calibration or validation of the model. If the behavioral requirements are defined in formal logic and the model is stochastic, verification or validation experiments using statistical model checking are suitable next experiments. If a measure of fulfillment of the requirement can be defined, calibration experiments are also possible.

Additional constraints may apply depending on the current life cycle phase, the properties of the simulation model, and the next goals. If the behavior of a model needs to be explored, a parameter scan might be suggested when at the beginning of a simulation study, whereas later a sensitivity analysis may be conducted on a reduced parameter set. If the simulation model has not yet been calibrated, a parameter estimation experiment will return not only parameter values but also their distribution. However, whether parameter estimation with methods like approximate Bayesian computation is applicable depends on the number of parameters to be fitted as well as the time needed to execute a simulation run. Furthermore, if data has already been used for calibration, the same data cannot be used in a simulation experiment with the goal validation.

What method to apply for that experiment type? For each type of simulation experiment different methods exist that impose specific constraints on their application. Only few of those methods might be available in the current modeling and simulation environment. Also the research objective of the simulation study can have an impact on the methods to be used. In [75], for example, different sensitivity analysis approaches are suggested, depending on whether the objective of the agent-based simulation study is more theoretical or more practical (and thus data-dependent) in nature. Additionally, properties of the simulation model may be the determining factor. For example, if there exists a linear relationship between the input factors and the output, the partial correlation coefficient (PCC) can be used as a method for global sensitivity analysis, whereas when the relationship is nonlinear but monotonic, an analysis based on the partial rank correlation coefficient (PRCC) can be conducted. However, there is a limitation to what can be done and known simply by looking at the simulation model specification and the conceptual model. In the context of simulation-based optimization, it was stated that the properties of the objective function determine which methods would be best, but these "are in general only known a posteriori" [156]. Consequently, instead of selecting the best method, the focus can be shifted to the problem of selecting a suitable method based on classifications (see Appendix B, [238]).

What parameters to choose? Even if, e.g., a sensitivity analysis experiment with a concrete method has been selected, and thus it is clear how the metamodel of that experiment looks like, still the different slots of the metamodel have to be filled with information. This includes information about how to access the model, the modeling and simulation environment, how to specify the time the simulation is run, the number of replications, or the confidence interval and the key variable to be observed. How to automatically parameterize some of these general inputs of a simulation experiment has already been discussed in Section 5.4.4. How to parameterize the experiment type- and thus method-specific part requires additional knowledge about methods and the meaning of their hyperparameters. In statistical model checking, for example, using the sequential probability ratio test [159] as a method requires explicit specification of the p-value and the indifference region δ, whereas other methods aim to compute p or adjust δ on-the-fly [432]. In sensitivity analysis, to give another example, the Method of Morris requires the number of levels per dimension as an input whereas others calculate those automatically given the total number n of design points.

What tools to use? For each experiment type or method, there exist a variety of tools—from free and open-source to proprietary, commercial software. Sensitivity analysis may, for example, be conducted in a general purpose language like Python, in a language for statistical computing like R, or a specialized tool such as GEM-SA, which uses Gaussian emulation and thus can handle large parameter spaces [202]. Consequently, a practical challenge in automatic experiment generation is selecting which tool or even combination of tools to use for specifying and conducting the experiment. When choosing a suitable tool, several aspects have to be considered that restrict the set of possible tools to use. Foremost, a tool is only suitable for generating the simulation experiment if it supports the chosen experiment type and method, and, ideally, if a binding to it has to be implemented in the experiment generator. Moreover, if the user has a preference for a specific tool (which can be derived from previous projects), it is likely that the user expects the same tool to be used again. Also, if the non-functional requirements of the simulation project ask for a specific type of license (e.g., a permissive license like the MIT license[69]), only a subset of tools offering the desired methods are allowed to be used. Additionally, if non-functional requirements demand runtime efficiency, benchmarks may be consulted to select the tool with the most efficient implementation.

What information to use and how to represent it? For collecting and representing context information about a simulation study, the definition of the conceptual model presented in Chapter 4 together with the PROV-DM can serve as a starting point. However, it needs to be inspected what information can be readily available for reuse during or across simulation studies, and what information is, on the other hand, so specific to one particular method and experiment that the modeler needs to provide it on demand. In the latter case, automating the generation of simulation experiments is hampered as users need to be asked to specify information explicitly and ideally formally. A balance has to be found that allows to generate as many simulation experiments specification as possible and as completely as possible, but that does not burden the modeler unnecessarily. To achieve this balance, inference rules need to be able to "read between the lines" of imperfect knowledge. The challenge becomes even bigger if the simulation study is conducted and documented in diverse, partly unstructured formats, and not as a provenance graph with explicit entities, activities, and dependencies. In practice, this may easily be the case if, e.g., the requirements, assumptions, etc. are described as semi-structured verbal narratives either in separate files [300], on wiki pages [83], or by using reporting guidelines [236]. Ideally, the experiment generation should adjust to the way and the environments in which simulation studies are naturally conducted and documented (e.g., increasingly electronic notebooks are used [433]). Thus, although the PROV-DM is recommended as a standard by the W3C, approaches for

[69] `https://opensource.org/license/MIT`, last accessed 19 July, 2024.

generating simulation experiments from scratch have to remain flexible to exploit alternative information sources and formats. In addition, they need to handle situations when there is no documentation at all or only partial documentation.

How can information about related simulation models help? Although the goal is to generate experiments "from scratch", the simulation study never starts at zero and then builds all knowledge from the ground up. Thus, one of the first things to consider are related simulation models. These can be models that are from the same domain, i.e., modeling the same system or a similar subsystem. If models are related with respect to the physical system they represent, simulation experiments for the new simulation model can be generated based on data stored with related models. Furthermore, there may be models related by the same modeling approach, language, or tooling that provide inspiration for the new project. To find related models, the various model libraries can be browsed and filtered by metadata about the content and semantics of the model (compare Section 4.3). The more metadata there is about what data, parameters, components, etc. were used to build the models, the better the (e.g., application-specific) constraints for their reuse can be evaluated.

How can general knowledge about modeling and simulation help? To generate an experiment from scratch, it is not sufficient to simply have the products of a simulation study available and accessible. Knowledge is also needed about how these products are related to each other, what the preconditions and constraints are for a simulation experiment, how this depends on the role and type of experiment, the goals and subgoals of the simulation study, and so on. For instance, general rules of thumb in modeling and simulation declare that calibration should always come before validation, and that calibration and validation should never use the same data. Furthermore, general knowledge of relevance to automatic experiment generation might be guidelines for working with stochastic simulations, or what it means to have a robust simulation model. Since this knowledge still lives mostly implicitly in the mind of the modelers and is passed on by text books and survey papers, further efforts have to be directed towards making these rules explicit in a computerized form.

How can application-specific knowledge help? Application-specific knowledge is instrumental in interpreting what the products of the simulation study are about and understanding what common challenges of that domain are. If the simulation model was, for instance, developed on a different temporal or spatial scale, or if it used data from a different cell line, species, or habitat, then additional simulation experiments will be required that ascertain the bias introduced by that choice of abstraction or data, and to establish trust and validity of the model. Application-specific knowledge (e.g., about recent findings, open research questions, or measurement data) may also help in setting up concrete simulation experiment specifications: to identify interesting what-if scenarios and to select the model parameters and outputs accordingly, or to suggest reasonable default values for the simulation stop time and observation interval.

How to account for project- or user-specific requirements? Most constraints regarding what to do next in the simulation study and what experiment to execute are based on provenance of the same or related studies, general knowledge and best practices as well as the application domain. Other constraints, however, may be motivated by the nature of the research project. If, e.g., the project requires certain analyses (for instance, uncertainty and sensitivity analyses) to be conducted for each model, and specific (possibly proprietary) tools for the analyses, or the strict usage of a scientific workflow is compulsory, then this affects the simulation experiments that can or need to be generated. On the other hand, modelers (i.e., users) of an experiment generation software may have personal preferences regarding how the generated code should look like, especially regarding what tools should be used. Furthermore, experienced users may

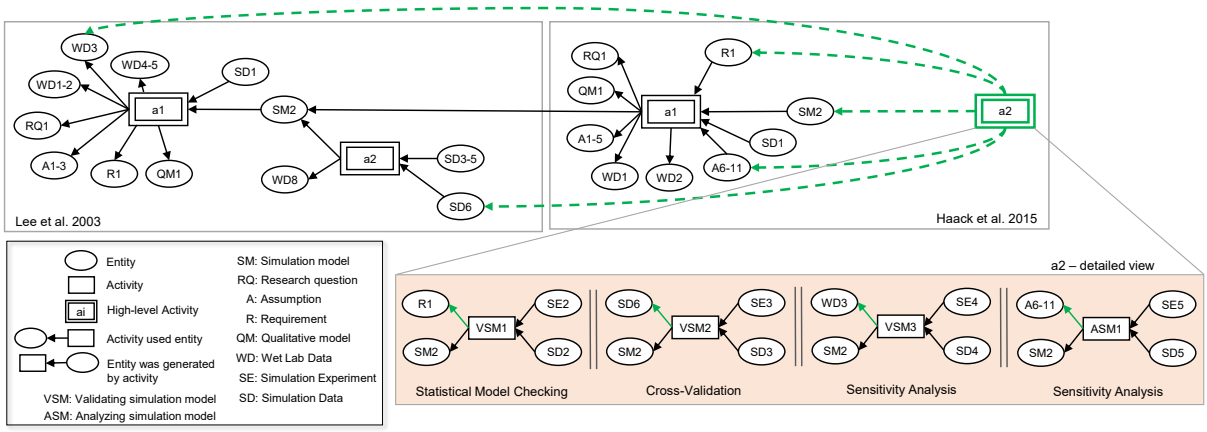


Figure 5.14: The provenance graphs of the Wnt simulation study and the previously conducted study by Lee et al. (top) relate different types of entities and activities. Four possible simulation experiments (bottom) are suggested for the hypothetical activity a2. For validation, a statistical model checking experiment based on a requirement, a cross-validation based on simulation data from a previous study, and a sensitivity analysis of wet-lab data could be conducted. In the case of model analysis, a sensitivity analysis based on assumptions is a suitable option. Figure adapted from Wilsdorf et al. [19].

have specific expectations for the methodologies to be used. E.g., what method to apply for calibration may be determined by previous experiences or by what is believed to be the current standard in the application domain. Thus, experiment generation should conveniently address these requirements.

5.8.2 Scenarios for Generating "from Scratch"

Figure 5.14 illustrates four exemplary rules for generating simulation experiments from scratch. The rules are again illustrated using the Wnt simulation study by Lee et al. [32] and Haack et al. [31]. The simulation studies are shown in aggregated form with the model building, calibration and analysis collapsed and with the artifacts of related simulation studies not depicted to keep the focus on the experiment generation. The first three rules assume that the user's intention is the validation of the simulation model. Naturally, if the user's intention was not model validation but instead, for example, the exploration of the model's behavior, sensitivity analysis, or predictive analysis, other types of experiments would be generated. Rule number four describes a scenario for sensitivity analysis.

The rules effectively exploit provenance information about previous, related simulation studies, and knowledge and expectations about the science and art of modeling and simulation itself. These expectations may refer to simulation studies more generally (e.g., to the role of requirements) or to application-specific information (e.g., the use of data stemming from different cell lines).

Statistical Model Checking based on Requirements What to do next depends on the objective of the simulation study, which is typically further substantiated by a set of requirements [2]. For example, the Wnt simulation model SM2 by Haack et al. is expected to reproduce wet-lab data (WD1). As part of the conceptual modeling, this wet-lab data was transformed into the formal requirement (R1) specified as a temporal logic formula. The requirement expects the concentration of β-catenin in the nucleus of the cell (β_{cat}) to peak within the first 40 to 120 minutes, which is reflected in the MITL [167] formula $\left[F_{[40,120]}\left(\beta_{\text{cat}} > 7500\right) \wedge G_{[120,t_{\text{end}}]}\left(\beta_{\text{cat}} < 6300\right)\right]$. Since neither WD1 nor R1 have yet been used for calibration in the current study, the requirement

can be used for validation. Leveraging an architecture like RASE, an experiment specification for a statistical model checking experiment (assuming a stochastic modeling approach has been chosen) can be semi-automatically generated and executed [237]. Thus, due to requirements being explicitly and formally specified, it can be automatically checked whether the simulation model fulfills or violates the requirements, e.g., with respect to data.

Cross-Validation with Data from Previous Simulation Studies Like many simulation models, the stochastic Wnt simulation model SM2 has not been developed from the ground up. It reused another simulation model developed by Lee et al. [32] and extended it by new submodules. If further information about the building process of the previous simulation model is available and requirements are formally specified, this information can be used to rerun experiments (e.g., from a1, a2 of the Lee et al. study) as a kind of regression testing with the new simulation model [229]. Rerunning these experiments independently reveals whether there are discrepancies in the behaviors of both models and provides valuable insights into the behavior of the new (and sometimes also the old) simulation model (see Section 5.6). Without previous simulation experiments accessible, further information about the preceding work on the Wnt signaling pathway needs to be exploited to automatically identify data that is suitable for cross-validating the simulation model. For instance, the simulation data SD6 may be reused from Lee et al.'s study, and a new validation experiment for the extended model SM2 may be generated and executed to compare the respective simulation data on β-catenin levels. This establishes a new provenance relation between the two studies, which can be seen in the hypothetical provenance graph of Figure 5.14.

Sensitivity Analysis of Wet-Lab Data from Previous Simulation Studies Based on the application domain, further simulation experiments can be suggested. For example, if parameters are taken as input from another model and that model, however, has been calibrated or validated based on wet-lab data from a different cell line, it makes sense to run a sensitivity analysis to assess the impact of these parameters on the simulation outcome. In our case, the model by Lee et al. has been developed based on data from Xenopus egg extracts (WD3), whereas the extended model has been based on data from human neural progenitor cells. Thus, a sensitivity analysis experiment should be run on the reused parameters to check whether a different parameter value derived from data for another cell line might have a significant impact on the simulation results, e.g., the initial β-catenin concentration or basal degradataion rate of β-catenin. Based on provenance and metamodels for sensitivity analysis, again this can be done semi-automatically [237].

Sensitivity Analysis based on Assumptions about Parameters Sensitivity analysis allows assessing how the variation in the model output can be apportioned to the different sources of variation, typically the input parameters. A sensitivity analysis experiment can serve different roles in a simulation study. If, e.g., one of the project requirements is to build a highly robust model, then sensitivity analysis would be conducted as part of the model validation. If, on the other hand, there is a lack of experimental data and the approach used for parameter fitting exhibits variability, an analysis experiment would be required to check the impact of the fitted (but still uncertain) parameters. In the Wnt simulation model, a sensitivity analysis could therefore focus on the six parameters that were introduced by the model extension and fitted based on wet-lab data. Since all other parameters of the model were obtained from previous model versions as well as from literature, these can be considered established. If the assumed parameter distributions as well as upper and lower bounds were recorded as explicit assumption artifacts in the provenance (A6-11) such an analysis could be generated semi-automatically [238].

5.9 Summary

The repetition of simulation experiments forms a salient feature of the model development process. Regression testing of successive model refinements within the same simulation study and cross-validation of related models from different simulation studies are just some examples of this. In this chapter, provenance information of simulation studies was exploited to automatically identify suitable experiments to reuse depending on the last model development steps and to automatically adapt the experiment specifications accordingly for the new model versions. The Reuse and Adapt framework for Simulation Experiments (RASE) hence is the first approach for reusing simulation experiments, in which historical context information is exploited and suitable next actions of the simulation study are triggered automatically. The central methods of the approach are the definition of provenance patterns and the construction of reuse rules. The collection of patterns and rules captures knowledge about simulation studies explicitly and thereby contributes to the conduction of more systematic and effective simulation studies. The applicability of the approach was successfully demonstrated in the context of human migration models as well as cell biology models. The software prototype of RASE is publicly and permanently available.

To expand the scope of the framework, it could be integrated, for example, with approaches for automatically analyzing and interpreting the simulation output [434], or approaches for experiment prioritization if limited resources are available, analogously to approaches for test prioritization known in software engineering [431]. Another important step forward will involve improving the API and adding (graphical) support for specifying new patterns and rules to allow for easier usability and customizability of the framework. In addition, more fine-granular provenance graphs may be required, if activities that go beyond the high-level goals of model calibration and validation need to be supported. With a refined provenance model, RASE may also be integrated more easily into current workflow systems [370, 52].

So far, the framework concentrates on the reuse of existing simulation experiments. Another interesting direction forward would be to exploit provenance for generating new experiments from scratch, i.e., without reusing a previous experiment specification as a blueprint. This might include incorporating more diverse, ideally standardized, formats for making context and other information explicit.

6 Conclusions

6.1 Summary

Simulation studies have long become an indispensable tool in both science and engineering. Conducting simulation studies in an effective and systematic manner is paramount for their success. However, specifying and executing simulation experiments is a complex task, in which the modeler needs to consider multiple factors. These include the current goal, the modeling and simulation approach being used, the application domain, the various experiment types, available methods, the software tools at hand, and the available knowledge sources that supply, e.g., theories and data.

A central question therefore arises: how can modelers be supported in conducting simulation experiments for calibration, validation, and analysis of simulation models, and how can this support be given automatically? One crucial strategy is the generation of simulation experiments by reuse and adaption. In this dissertation, three key ingredients of such an approach have been identified and studied.

The first key ingredient involves the explicit and semi-automatic specification of simulation experiments. To facilitate this, a model-driven engineering (MDE) framework was developed. This framework provides means for the specification and composition of metamodels, a metamodel repository, a variety of tool bindings with bidirectional code transformations, and a graphical user interface (GUI) as well as a command-line interface (CLI) for seamless interaction and integration with the approach. The MDE-based framework improves knowledge structuring and sharing within but also across the different simulation domains and approaches by establishing a common language via metamodels. Metamodels play a crucial role in expressing the central concepts for various modeling and simulation approaches and experiment types. The on-demand composition of the so-called base metamodels and metamodels for the diverse experiment types guides the modeler in specifying intricate simulation experiments. Thereby, the approach enhances the productivity of modelers and the quality of simple and complex simulation experiments. Furthermore, the framework facilitates the automatic reuse of simulation experiments and the automatic generation and execution of simulation experiments as part of workflow environments.

The second key ingredient is context about the simulation study, and particularly simulation experiments. Context is key in deciding which simulation experiment to execute, when, and how. Various approaches for representing and exploiting context of a simulation study were explored. A comprehensive and partly formalized definition of the conceptual model was introduced, which encompasses early-stage artifacts of the types research objectives, requirements, methodologies, assumptions, qualitative models, and data and information sources. The definition includes partly formalized metadata for each artifact via domain-specific languages and ontologies. Additionally, provenance aims to narrate the entire story of a simulation study by relating conceptual model artifacts to primary artifacts like simulation models, simulation experiments, and simulation data. The provenance data model (PROV-DM) relates artifacts via activities and directed edges forming a directed acyclic graph. Various kinds of graph queries can be executed on this representation to gain insights into the simulation study as well as its related studies, and to extract the information required for enabling reproducibility and reusability. Moreover, open model databases were investigated as a key means for distributing, editing, and interconnecting provenance graphs and the conceptual model, ensuring their computational accessibility.

The third key ingredient is the Reuse and Adapt framework for Simulation Experiments

(RASE). It integrates the explicit context information given by provenance and the conceptual model, along with explicit experiment specification and model-driven code generation. For the integration, a third ingredient is crucial: provenance patterns and inference rules defined based on these patterns. They enable the automatic initiation of an experiment generator depending on the last activities of a modeler, automatic identification of earlier simulation experiments to reuse, and their adaption according to the context of a new simulation model. RASE thus supports the salient iterative feature of the modeling and simulation life cycle in an automated manner. Automatic regression testing of successive model refinements and cross-validation of related models are just some examples of this. The extensibility and customizability of the framework ensure that it can effortlessly accommodate a variety of scenarios, providing support for a wide range of simulation studies.

The approaches presented in this dissertation support modelers in conducting their simulation experiments, as well as their entire simulation studies, more systematically, effectively, and with methodological rigor. At the same time, the user's flexibility and control are maintained by providing the necessary user interfaces and feedback loops. The various benefits were demonstrated in open-source software prototypes and a variety of case studies from different application domains, also using various modeling and simulation approaches. These included a stochastic discrete-event simulation of a cell signaling pathway, virtual prototyping of a neurostimulator, finite element analysis of electric fields, and agent-based simulation of human migration.

The results of this dissertation open up new avenues for further research towards the automation of simulation experiments and simulation studies. Particularly, the central premises "the past informs the future" and "from context comes the understanding of everything" are not only true for the reuse and adaption of simulation experiments but may also be applied for generating new simulation experiments from scratch, or in the bigger picture, even for generating simulation models.

6.2 Outlook

Along the way, ideas for extending the proposed frameworks, various gaps in existing research, and avenues for future work were identified in the chapter summaries. These can be grouped into four research areas:

Automating the M&S Life Cycle This dissertation focused on the reuse and adaption of simulation experiments. A considerable step forward will be the generation of simulation experiments from scratch without directly building on top of previous simulation experiments of the same or a related study. Here, the various experiment types and the different purposes they have in the M&S life cycle need to be considered. So far, the focus was on experiments for calibration, validation, and analysis of the simulation model. However, the approach may also be used for generating simulation experiments for performance evaluation of different simulation algorithms. Either way, the challenge will be to correctly recognize user intent to generate the best simulation experiments for a given situation.

Extensions of the framework may also be applied for supporting the other stages of a simulation study. For example, modelers may want automatic support in building the conceptual model, and for making the next modeling decisions. Consequently, frameworks for automatically generating simulation models, including the automatic reuse and composition of models, need to be developed. In addition, explicit artifacts of the conceptual model, such as requirements and assumptions, may be generated by taking into account other parts of the simulation study's context. There already exist approaches that can generate model equations that best fit some measurement data using symbolic regression [435], or approaches that can generate software requirements from verbal narratives [436].

However, fully automating the M&S life cycle would require automatic interpretation of the various artifacts, including the simulation data, and drawing consequences for the next steps based on the knowledge, expertise, and preferences of the modeler. Therefore, scenarios where the human modeler is kept in the loop seem promising. Research in digital twins, for instance, is already addressing the problems of how to enable users to quickly understand the model outputs, run predictions for specific scenarios, and quantify the level of confidence in the results [437]. In addition, key questions are determining which level of autonomy to grant the digital twin and identifying the cognitive tasks it should be responsible for.

Establishing Languages and Standards Efforts within the various modeling and simulation communities have primarily concentrated on formal specification languages for simulation experiments and simulation models [36, 35, 395]. The demand for domain-specific languages persists, particularly for making the various artifacts of the conceptual model explicit along with their metadata. For instance, despite the existence of formal approaches like temporal logics for expressing requirements ([167, 170]), those still need to be tailored into internal or external domain-specific languages for the diverse types of requirements used in simulation studies and the various application domains. Furthermore, methods for specifying the qualitative model may need to be tailored. For instance, domain-specific causal loop diagrams have recently been discussed, which incorporate (ontological) annotations [438]. Additionally, languages for describing assumptions necessitate further exploration.

Furthermore, formal representations are required for general methodological knowledge about goals and subgoals, properties of experiment types and methods, metadata about tools, as well as the overarching rules about the M&S life cycle. Moreover, to be intelligible and easy-to-use, tailoring languages to the nuances of their specific application domains is crucial. Most importantly, for ensuring the unambiguity of specifications, ontologies should be used for semantic annotations, especially when concepts are employed across diverse application domains, simulation approaches, and tools. Knowledge graphs are a powerful instrument for structuring the various concepts, their attributes, and relationships [439].

Further standardization efforts will be required, but these need to be community-driven so that all domain-specific intricacies can be taken into account. The standardization processes may be accompanied by existing initiatives, such as SISO[70], COMBINE[71], or OMF[72].

In addition to language development, enhanced tool support for specifying knowledge is required, including means for integration with established frameworks and knowledge bases. Finally, maintaining and improving the necessary tools should become a priority. This extends to provenance, documentation standards, and open repositories. Their increasing role for modeling and simulation studies was demonstrated and discussed in this dissertation.

Acquiring Knowledge and Integrating Sources Effectively generating simulation experiments, models, or other artifacts depending on context requires an abundance of explicitly and unambiguously specified knowledge. Various categories of knowledge were discussed in this dissertation, encompassing general knowledge about the M&S process, knowledge about methodologies and tools, and the intricacies of the application domains. Unfortunately, most of this knowledge is not explicitly and machine-accessibly available.

Consequently, approaches for knowledge acquisition will be crucial for building dedicated support for simulation studies. Methods for manual collection include expert interviews in the domains, literature studies, or large scale surveys. Automatic extraction of knowledge from unstructured or semi-structured text may be facilitated by various natural language processing

[70] Simulation Interoperability Standards Organization, `https://www.sisostandards.org/`, last accessed 19 July, 2024.
[71] COmputational Modeling in BIology NEtwork, `https://co.mbine.org/`, last accessed 19 July, 2024.
[72] Open Modeling Foundation, `https://www.openmodelingfoundation.org/`, last accessed 19 July, 2024.

techniques, including recent generative artificial intelligence models [440]. Advances in machine learning (ML) methods also seem promising, e.g., for mining process knowledge [441].

Additionally, further tool support is necessary to incentivize users to manually collect detailed provenance information. Another option are automatic approaches that are able to capture provenance in a transparent and light-weight manner that does not burden the modeler [300]. However, seamlessly integrating these approaches with the diverse working environments of modelers remains a challenge.

The sources for automatic knowledge extraction include knowledge graphs, open databases, research artifacts, code repositories, provenance graphs, and scientific literature. Typically, a combination of these sources is required to address specific questions or tasks. Managing and exploiting information from these diverse sources is an evolving challenge that has recently been tackled by "data fabrics" [442], flexible software architectures that allow integrating heterogeneous data sources. Similar architectures may be facilitated for integrating knowledge in modeling and simulation.

Applying Machine Learning The rule-based and knowledge-based approach for generating simulation experiments presented in this dissertation has several key advantages. First, various knowledge that is of importance in successfully conducting simulation studies is made explicit, and may be reused and extended in future endeavors to support simulation studies. Second, concrete simulation studies are thoroughly documented, including explicit experiment specifications, conceptual model, and provenance of the entire study. This will be advantageous for the various scientific communities as it lends credibility, traceability, and reusability to their simulation studies. Third, the defined patterns and rules enable consistency checks for a variety of simulation studies.

As the framework grows in complexity, more applications shall be supported, and more data become available, the various tasks of the experiment generator may be increasingly based on ML. However, for ML approaches to be viable, data needs to be available and adequately represented for training, testing, and validating the ML models. Currently, data are sparse, and widespread usage of formal provenance models and open modeling repositories for telling the whole story of a simulation study is just starting to become indispensable.

Nevertheless, in the pursuit of an intelligent M&S life cycle, ML methods have made progress. But so far, the different tasks of a modeler have been targeted in isolation, e.g., using reinforcement learning or decision trees for choosing the right methods for a simulation experiment [110], or using deep neural networks and optimization for selecting hyperparameters [443]. Combining and developing methods to support the bigger picture remains a topic of future research.

All in all, the methods developed in this dissertation contribute towards making the context of simulation studies explicit and exploiting this information automatically to generate experiment specifications in both tool-independent and tool-dependent manners. Automated situation-specific reuse and adaption of experiments during the entire simulation study was achieved. However, fully automatic reuse and adaption of simulation experiments beyond the supported scenarios, as well as the fully automatic generation of experiments from scratch, can only be realized with further explicit context information and suitable methods for its interpretation. Therefore, progress in these four areas of future research will greatly benefit automatic experiment generation and the broader automatic support of simulation studies. Addressing these topics in the near future will also drive the field of modeling and simulation forward, positioning it at the forefront of current developments in computer science, such as those in artificial intelligence.

A Metamodels of Simulation Experiments

In Chapter 3, metamodels were introduced as the central approach for representing the ingredients of diverse simulation experiments. First, base metamodels were employed to represent the simulation experiments within a specific modeling and simulation approach. Subsequently, experiment type metamodels were utilized to represent, for instance, the structure of global sensitivity analysis. Through composition of these two types of metamodels, complex simulation experiments could be constructed. The following tables present additional metamodels.

Table A.1 depicts the base metamodel for

- finite element analysis,

which was discussed in Section 3.5.1.

Tables A.2-A.9 show the metamodels for the experiment types

- steady state estimation,
- parameter scan,
- local sensitivity analysis,
- parameter estimation,
- optimization,
- statistical model checking,
- what-if analysis,
- and convergence testing.

Note that these metamodels serve as an initial draft. They can be flexibly extended to support additional, more specialized simulation experiments and methods.

In the tables, the same symbols as in Chapter 3 will be used. The rows describe the different input properties of the metamodel. Sub-properties are indented. Properties of type Map consist of a key ("κ") and values ("ν"). Alternative metamodel parts are indicated by "|".

Table A.1: Base metamodel for conducting simulation experiments in finite element analysis. A geometric model, a physical model, and a solver need to be provided and configured. Optionally, observations can be specified. The metamodel may be extended, e.g., with further types of boundary condition, such as the von Neumann boundary condition. EQS – Electro Quasi Statics, MUMPS – MUltifrontal Massively Parallel sparse direct Solver, CG – Conjugate Gradient method, XDMF – eXtensible Data Model and Format.

	Name	Description	Type	Choices	Required
Geometric Model	studyName	Name of the simulation study	String	–	no
	dimensions	Number of dimensions	Integer, >0, ≤ 3	–	yes
	geometry	Choose geometry type	Alternative	–	yes
	| meshFile	Path to mesh file	String	–	yes
	| geometryFile	Path to geometry file	String	–	yes
	geometryValues	Define the geometry	Map	–	no
	κ subDomain	Geometry element	String	–	no
	ν subDomainValue	Value of geometry element	Real	–	no
Physical Model	physics	Choose a physical model	String	EQS, ...	yes
	materialProperties	Define materials and their properties	Map	–	yes
	κ materialName	Material used in the model	String	–	yes
	ν conductivityValue	Conductivity of the material	Real	–	yes
	ν permittivityValue	Permittivity of the material	Real	–	yes
	boundaryCondition	Choose a boundary condition	Alternative	–	yes
	| dirichlet	Dirichlet boundary condition	Map	–	yes
	κ boundary	Name of the boundary	String	–	yes
	ν boundaryValue	Boundary value	Real	–	yes
	| vonNeumann	Von Neumann boundary condition	Map	–	–
	...	...	...	...	...
	frequency	Frequency value	Real	–	yes
Simulation	solver	Choose a solver type	String	MUMPS, ...	yes
	element	Choose an element type	String	CG, ...	yes
	degree	Degree of the polynomial	Integer	–	yes
	meshRefinement	Use iterative mesh refinement	Map	–	no
	κ refinedSubDomain	Name of the subdomain to be refined	String	–	no
	ν refinementCycles	Number refinement steps	Integer	–	no
Observation	observationPoints	Points at which equations are solved	Map	–	no
	κ observationPosition	Coordinates of the observed point	Array<Real>, length $\in \{1..3\}$	–	no
	ν observationAlias	Alias for the observation	String	–	no
	outputFormat	Choose an output format	String	XMDF, ...	no
	observables	Calculation of observed properties	Map	–	no
	κ observableName	Name of the observable	String	–	no
	ν observableExpression	Expression for calculating the observable	String	–	no

Table A.2: Metamodel for the experiment type "steady state estimation" (SE) w.r.t. the observation specified in the base experiment. This is the most general representation requesting simply the name of the method and information about the method's parameters. To lend further support for users, it may be refined with details about the specific methods.

	Name	Description	Type	Choices	Default	Required
Steady State Estimation	methodName	Name of the SE method	String	–	–	yes
	methodParameters	Parameters of the SE method	Map	–	–	no
	κ methodParameterName	Name of the parameter	String	–	–	no
	ν methodParameterValue	Value of the parameter	Any	–	–	no

Table A.3: Metamodel for the experiment type "parameter scan". A component for providing factor information and a component with two alternative experiment designs are outlined. Other designs may be added, e.g., the Central Composite Design [231]. Furthermore, the metamodel may be extended to include categorical variables.

	Name	Description	Type	Choices	Default	Required
Factor Information	factors	Information about the model factors	Map	–	–	yes
	κ factorName	Name of the factor	String	–	–	yes
	ν factorMinimumValue	Lower bound on the factor value	Real	–	–	yes
	ν factorMaximumValue	Upper bound on the factor value	Real	–	–	yes
	ν factorDistribution	Assumed distribution of the factor	String	Uniform, Normal, ...	Uniform	yes
	ν factorDistributionParameters	Parameterize the distribution	Map	–	–	no
	κ distributionParameterName	Parameter of the distribution	String	–	–	no
	ν distributionParameterValue	Initialize the distribution parameter	Real	–	–	no
Experiment Design	designMethod	Choose the experiment design	Alternative	–	–	yes
	| fullFactorialScan	Full factorial design	–	–	–	yes
	numberOfLevels	Levels per factor	Integer	–	–	yes
	| latinHypercube	Latin hypercube design	–	–	–	yes
	sampleSize	Number of samples	Integer	–	–	yes
	isOrthogonal	Use an orthogonal Latin hypercube	Boolean	–	–	no
	| ...	...	...	...	...	...
	seed	Random seed for reproducibility	Integer	–	–	no

Table A.4: Metamodel for the experiment type "local sensitivity analysis" based on a two-level full factorial design. It may, e.g., be extended with a module for visualizing the results of a sensitivity analysis [125, Appendix A].

	Name	Description	Type	Choices	Default	Required
Two-Level Full Factorial Design	factors	Information about the model factors	Map	–	–	yes
	κ factorName	Name of the factor	String	–	–	yes
	ν factorBaseCase	Base value of the factor value	Real	–	–	yes
	ν factorSensitivityCase	Sensitivity value of the factor value	Alternative	–	–	yes
	| absoluteValue	Absolute value	Real	–	–	yes
	| percentageChange	Percentage change	Real	–	–	yes
Indices	individualEffects	Calculate the individual (main) effects	Boolean	–	true	no
	totalEffects	Calculate the total effects	Boolean	–	–	no
	interactionEffects	Calculate the interaction effects	Boolean	–	–	no

Table A.5: Metamodel for the experiment type "parameter estimation". It may be extended with a visualization component, as suggested by PEtab [206].

	Name	Description	Type	Choices	Default	Required
Prior Parameter Information	parameters	Information about the model parameters	Map	–	–	yes
	κ parameterName	Name of the parameter	String	–	–	yes
	ν priorDistribution	Assumed distribution type	String	...	Uniform	yes
	ν priorDistributionParameters	Parameterize the distribution	Map	–	–	no
	κ priorDistributionParameterName	Parameter of the distribution	String	–	–	no
	ν priorDistributionParameterValue	Initialize the distribution parameter	Real	–	–	no
Parameter Estimation	measurementFile	Observed data provided as a file	String	–	–	yes
	distanceFunction	Distance function btw. simulated and measured data	String	–	–	yes
	methodName	Method used for parameter estimation	String	–	–	yes
	methodParameters	Parameters for the method (e.g, ϵ)	Map	–	–	no
	κ methodParameterName	Name of the parameter	String	–	–	no
	ν methodParameterValue	Value of the parameter	Any	–	–	no

Table A.6: Metamodel for the experiment type "optimization". As methods are quite different, their parameters can be flexibly added. Later, specialized metamodels may be added, e.g., for simulated annealing, or to support multi-objective optimization [444].

	Name	Description	Type	Choices	Default	Required
Parameter Information	parameters	Information about the parameters	Map	–	–	yes
	κ parameterName	Name of the parameter	String	–	–	yes
	ν parameterMinimumValue	Lower bound on the parameter value	Real	–	–	yes
	ν parameterMaximumValue	Upper bound on the parameter value	Real	–	–	yes
	ν parameterInit	Initial value for the parameter	Real	–	–	yes
Optimization Method	objectiveFunction	Objective function	String	–	–	yes
	minimize	Whether to minimize the function	Boolean	–	true	yes
	methodName	Optimization method	String	Simulated Annealing, ...	–	yes
	methodParameters	Parameters of the optimization method	Map	–	–	no
	κ methodParameterName	Name of the method parameter	String	–	–	no
	ν methodParameterValue	Value of the method parameter	Any	–	–	no

Table A.7: Metamodel for the experiment type "statistical model checking". Extensions of the metamodel may include explicit information about how the specified property is checked. LTL – Linear Temporal Logic, MITL – Metric-Interval Temporal Logic, SSP – Single Sampling Plan, SPRT – Sequential Probability Ratio Test.

	Name	Description	Type	Choices	Default	Required
Property	propertyType	Language for specifying the property	String	LTL, MITL, ...	–	yes
	propertySpecification	Property to be evaluated	Alternative	–	–	yes
	| propertyExpression	Property as expression	String	–	–	yes
	| propertyFile	Property provided as file	String	–	–	yes
Hypothesis Test	testMethod	Name of the Statistical test	String	SSP, SPRT, ...	–	yes
	p	Confidence level	Real $\in [0,1]$	–	–	no
	alpha	Probability of type I error	Real $\in [0,1]$	–	–	yes
	beta	Probability of type II error	Real $\in [0,1]$	–	–	yes
	delta	Half-width of the indifference region	Real $\in [0,1]$	–	–	yes

Table A.8: Metamodel for the experiment type "what-if analysis". Extensions may define policy interventions explicitly as part of the simulation experiment or include analysis and visualization of the results.

	Name	Description	Type	Choices	Default	Required
What-if Scenarios	scenarios	Model variants (structures) to explore	Map	–	–	yes
	κ scenarioName	Name of the scenario	String	–	–	yes
	ν modelFile	Specify the simulation model	Alternative	–	–	yes
	| folder	Folder of the simulation model	String	–	–	yes
	fileName	Name of the simulation model	String	–	–	yes
	| reference	Reference to the simulation model	String	–	–	yes
	ν parameters	Parameterize this model	Map	–	–	no
	κ parameterName	Name of the parameter	String	–	–	no
	ν parameterValue	Set value of the parameter	Real	–	–	no

Table A.9: Metamodel for the experiment type "convergence testing". Currently, adaptive refinement of a finite element mesh is considered. In the future, convergence of the numerical solver w.r.t. the step size may be included.

	Name	**Description**	**Type**	**Choices**	**Default**	**Required**
Adaptive Mesh Refinement	region	Region of interest	String	–	–	yes
	errorMetric	Function for estimating the discretization error	String	–	–	yes
	stopCriterion	Choose type of stop criterion	Alternative	–	–	yes
	|maximumInterations	Maximum number of mesh refinement steps	Integer	–	–	yes
	|errorThreshold	Error threshold for mesh refinement	Real	–	–	yes
	minimumElementSize	Minimum size of the finite elements (initial meshing hypothesis)	Real	–	–	yes
	maximumElementSize	Maximum size of the finite elements (initial meshing hypothesis)	Real	–	–	yes

B Excursus: Ontology of Sensitivity Analysis Methods and its Use for Method Selection

The broader context of a simulation study must consider the methodologies employed. This entails assessing the types of experiments used, the reasons behind their selection, and how they fit into the context of a given simulation model and its history. In particular, understanding the properties of a model and its development process that allow for the application of specific methods is vital for steering the next actions in a simulation study effectively.

Third, to address the challenge of selecting appropriate experiment types and methods within a specific context, ontologies and best practice rules are defined and utilized. By explicitly representing methodological aspects and relating them to characteristics of the simulation study, semi-automatic decisions can be made. This section summarizes parts of the paper [238, © 2021 IEEE.].

With regards to simulation experiments, which method to use is a frequently asked question [445]. For instance, the automatic selection of methods in modeling and simulation has been investigated with the goal of achieving more robust results or faster runtimes [110]. However, for experiment types such as optimization, finding the best performing method proved difficult [156], due to the fact that properties of the response surface are typically "not known a priori". Another problem is that, with regards to the applicability of a method, there may not be one best method but rather various alternatives. Modelers may not be aware of them or their differences. Also, methods are often applied without considering the context. A recent review on sensitivity analysis (SA) found that approximately half of the conducted analyses were applied falsely [126]. This makes it difficult to learn from previous cases.

Therefore, it is crucial that support for SA (as well as other critical experiment types) relies on thoroughly collected knowledge based on theory instead of imitating previous applications and carefully consider the context of the simulation study. Providing a knowledge base that can guide the decision-making process by eliminating certain methods based on provided criteria, could help to bundle expertise and automate certain steps, which eventually prevents fundamental methodological mistakes and gives some guidance, especially for inexperienced users.

Here, ontologies are used to assign properties to methods, e.g., which types of models they can or cannot handle, how computationally expensive they are, or which kinds of measures can be calculated (e.g., first-order, second-order sensitivity indices). Listing B.1 shows an excerpt of an ontology developed specifically for SA methods [238]. It is written in the Manchester OWL syntax [446] and describes the properties of the Elementary Effects method, also known as the Morris method. Given an ontology, a list of suitable methods can be extracted by building the right description logic (DL) query.

For instance, one best practice about sensitivity analysis from the literature states: *"Both uncertainty and sensitivity analysis should be based on a global exploration of the space of input factors, be it using an experimental design, Monte Carlo or other ad-hoc designs. [...] local/OAT methods do not adequately represent models with nonlinearities"* [126]. This best practice promotes the use of global SA methods over local SA methods, especially for nonlinear models. In the case where the model response is assumed nonlinear or no such information is found, only global SA methods are selected from the ontology, using the query: `SAMethod and GlobalMethod`. If the model is assumed to exhibit linear behavior, also local one-at-a-time (OAT) methods are selected: `SAMethod and (GlobalMethod or LocalMethod)`.

Another best practice rule states: *"When sensitivity analysis is performed, it should allow the*

relative importance of input factors and combinations of factors, to be assessed, either visually (scatterplots) or quantitatively (regression coefficients, sensitivity measures or other)" [126]. The following DL query can filter all methods that can be used for factor ranking and can calculate the interaction effects between parameters: `has Purpose some Ranking and hasSpecialty Interactions`.

```
Class: Morris
SubClassOf:
SAMethod
GlobalMethod
hasComplexity some LowComplexity
hasPurpose some Ranking
failsWith some NonSmoothModelResponse
...
DisjointWith:
FAST, Sobol, SRCs, ...
```

Listing B.1: Specification of the class "Morris" of the SA method ontology. Reprinted from [238]. © 2021 IEEE.

The developed SA ontology was implemented using Protégé [447]. Figure B.1 shows a screenshot of the Protégé frontend displaying the classes of the SA ontology. It was based on the categorizations obtained from two review papers [125, 127]. For the concrete methods, a few methods were pre-selected that were found readily available in popular SA packages [448, 449]. The ontology therefore currently focuses on the following methods: a simple two-level full factorial analysis as a representative of local SA, and the following global methods: the Elementary Effects method by Morris, Delta, Sobol, RBD-FAST, FAST, SRC, and PRCC. Note that this list is not complete and that the ontology could be extended with additional methods.

To put the ontology to work in an existing framework for conducting simulation experiments, it can be exported/imported via the OWL (Web Ontology Language) and queried, e.g., using the OWL Java API [450]. Through this API, DL queries can be submitted, e.g., to the Hermit reasoner [451].

Now given a simulation model and some context information, a query may generated to extract all suitable methods from the ontology of SA methods that are applicable in the provided context. Listing B.2 shows the generated DL query and the query result for a Wnt signaling model. Details of the case study are given in [238].

```
Query: GlobalMethod and hasPurpose some Ranking and
hasSpecialty some Interactions
Result: FAST, Morris, Sobol
```

Listing B.2: DL query for finding all suitable SA methods for the current simulation model. Reprinted from [238]. © 2021 IEEE.

In conclusion, automatically generating and executing simulation experiments holds the promise of facilitating the conduction of more consistent, systematic, and efficient simulation studies. One building block in this endeavor is choosing a suitable experimentation method for a particular experiment type (e.g., sensitivity analysis, simulation-based optimization, etc.). However, which methods are suitable, and how they have to be parametrized and executed largely depends on the context in which the experiment is conducted.

So far, the knowledge about the various methods is extracted manually from literature reviews. In the future, obtain the data (i.e., the ontologies and rules) via machine learning or based, which, however, require huge amounts of training data. But with the recent emergence of large language models, automatic ontology construction now seems in reach [452].

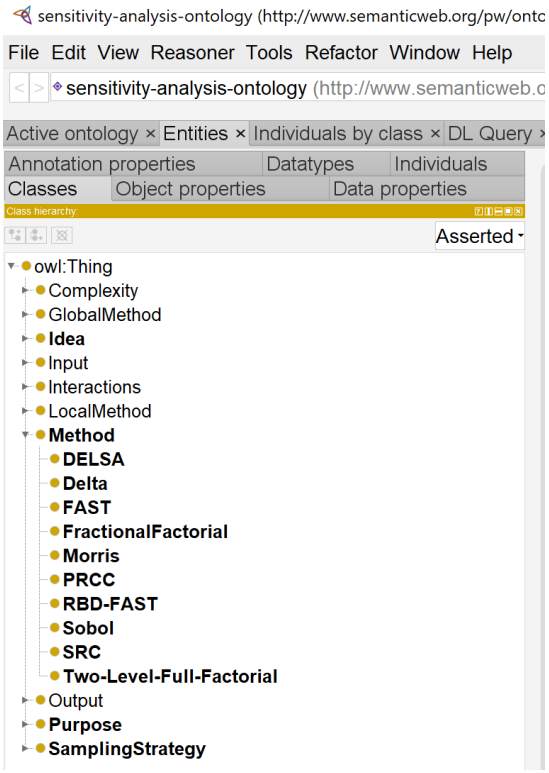


Figure B.1: Ontology of sensitivity analysis methods implemented with Protégé.

C Statistical Model Checking Experiment in SESSL

```
import sessl._
import sessl.mlrules._
import sessl.verification._
import sessl.verification.mitl._

// Minimum and maximum values of beta catenin (from Lee et al. 2003)
val maxR = 10000
val minR = 7500
// Observed time range
val startObsTime = 300
val stopObsTime = 420

// Statistical model checking
execute(
  new Experiment with Observation with ExpressionObservation
  with StatisticalModelChecking with ParallelExecution with CSVOutput {
    // Set up the model
    model = "../models/M2_2.mlrj"
    // Set initial species count and experiment specific reaction rate coefficient
    set("nWnt" <~ 300)
    set("scalingFactor" <~ 0.28)

    // Set up the simulator
    simulator = SimpleSimulator()
    parallelThreads = 5

    // Set up the observation
    stopTime = stopObsTime
    observeAt(range(startObsTime, 10, stopObsTime))
    val bCat = observe("bcat" ~ count("Cell/Nuc/Bcat"))

    // Initialize the model checker
    test = SequentialProbabilityRatioTest(p = 0.8, alpha = 0.05, beta = 0.05, delta = 0.05)
    // Property to be checked
    prop = (_, outputs) => {
      val values = outputs(bCat).asInstanceOf[Trajectory[Double]].toMap
      values.forall {case (t,y) => (y < maxR && y > minR)}
    }
    // Qualitative result
    withCheckResult { result =>
      println(s"Satisfaction status: ${result.satisfied}")
    }
    // Store results of the runs
    withExperimentResult(writeCSV)
  }
)
```

Listing C.1: Statistical model checking experiment in SESSL for cross-validation of the Lee et al. (2003) and Haack et al. (2015) models. Using the sequential probability ratio test, the stochastic output trajectories of the extended model are assessed to determine whether the species count of β-catenin consistently falls within the range (7500,10000) in the time interval [300,420].

Bibliography

[1] E. Winsberg. *Science in the age of computer simulation*. University of Chicago Press, 2010.

[2] O. Balci. A life cycle for modeling and simulation. *SIMULATION*, 88(7):870–883, 2012. doi:10.1177/0037549712438469.

[3] R. G. Sargent. Verification and validation of simulation models. *Journal of Simulation*, 7 (1):12–24, 2013. doi:10.1057/jos.2012.20.

[4] A. M. Law. How to build valid and credible simulation models. In *2019 Winter Simulation Conference (WSC)*, pages 1402–1414. IEEE, 2019. doi:10.1109/WSC.2005.1574236.

[5] S. Robinson. *Simulation: the practice of model development and use*. John Wiley & Sons, Inc., 2004. ISBN 978-0-470-84772-5.

[6] R. E. Shannon. Introduction to the Art and Science of Simulation. In *Proceedings of the 30th Conference on Winter Simulation*, WSC '98, pages 7–14, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press. ISBN 0-7803-5134-7.

[7] Wissenschaftsrat. *Bedeutung und Weiterentwicklung von Simulation in der Wissenschaft - Positionspapier*. Drs. 4032 - 14. `www.wissenschaftsrat.de/download/archiv/4032-14.pdf`, 2014. Last accesssed 19 July, 2024.

[8] R. Fujimoto, C. Bock, W. Chen, E. Page, and J. H. Panchal, editors. *Research Challenges in Modeling and Simulation for Engineering Complex Systems*. Springer Cham, 2017. doi:10.1007/978-3-319-58544-4.

[9] N. I. Sarkar and J. A. Gutierrez. Revisiting the issue of the credibility of simulation studies in telecommunication networks: highlighting the results of a comprehensive survey of IEEE publications. *IEEE Communications Magazine*, 52(5):218–224, 2014. doi:10.1109/MCOM.2014.6815915.

[10] D. Kluess, E. Soodmand, A. Lorenz, D. Pahr, M. Schwarze, R. Cichon, P. A. Varady, S. Herrmann, B. Buchmeier, C. Schröder, S. Lehner, and M. Kebbach. A round-robin finite element analysis of human femur mechanics between seven participating laboratories with experimental validation. *Computer Methods in Biomechanics and Biomedical Engineering*, 22(12):1020–1031, 2019. doi:10.1080/10255842.2019.1615481.

[11] S. J. E. Taylor, T. Eldabi, T. Monks, M. Rabe, and A. M. Uhrmacher. Crisis, what crisis: Does reproducibility in modeling & simulation really matter? In *Proceedings of the 2018 Winter Simulation Conference*, pages 749–762. IEEE, 2018. doi:10.1109/WSC.2018.8632232.

[12] A. Saltelli and S. Funtowicz. What is science's crisis really about? *Futures*, 91:5 – 11, 2017. doi:10.1016/j.futures.2017.05.010.

[13] C. Di Ciccio, A. Marrella, and A. Russo. Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. *Journal on Data Semantics*, 4: 29–57, 2015. doi:10.1007/s13740-014-0038-4.

[14] F. J. Montáns, F. Chinesta, R. Gómez-Bombarelli, and J. N. Kutz. Data-driven modeling and learning in science and engineering. *Comptes Rendus Mécanique*, 347(11):845–855, 2019. doi:10.1016/j.crme.2019.11.009.

[15] F. E. Cellier. *Continuous System Modeling*. Springer Science & Business Media, 2013. doi:10.1007/978-1-4757-3922-0.

[16] S. Rybacki, S. Leye, J. Himmelspach, and A. M. Uhrmacher. Template and frame based experiment workflows in modeling and simulation software with worms. In *2012 IEEE Eighth World Congress on Services*, pages 25–32, 2012. doi:10.1109/SERVICES.2012.22.

[17] P. Wilsdorf, J. Heller, K. Budde, J. Zimmermann, T. Warnke, C. Haubelt, D. Timmermann, U. van Rienen, and A. M. Uhrmacher. A model-driven approach for conducting simulation experiments. *Applied Sciences*, 12(16), 2022. doi:10.3390/app12167977.

[18] A. Carusi, K. Burrage, and B. Rodríguez. Bridging experiments, models and simulations: an integrative approach to validation in computational cardiac electrophysiology. *American Journal of Physiology-Heart and Circulatory Physiology*, 303(2):H144–H155, 2012. doi:10.1152/ajpheart.01151.2011.

[19] P. Wilsdorf, F. Haack, K. Budde, A. Ruscheinski, and A. M. Uhrmacher. Conducting systematic, partly automated simulation studies – unde venis et quo vadis. In *17th International Conference of Numerical Analysis and Applied Mathematics*, 2020. doi:10.1063/5.0026939.

[20] D. Groen, A. P. Bhati, J. Suter, J. Hetherington, S. J. Zasada, and P. V. Coveney. FabSim: Facilitating computational research through automation on large-scale and distributed e-infrastructures. *Computer Physics Communications*, 207:375–385, 2016. doi:10.1016/j.cpc.2016.05.020.

[21] T. Kiss, J. DesLauriers, G. Gesmier, G. Terstyanszky, G. Pierantoni, O. A. Oun, S. J. Taylor, A. Anagnostou, and J. Kovacs. A cloud-agnostic queuing system to support the implementation of deadline-based application execution policies. *Future Generation Computer Systems*, 101:99–111, 2019. doi:10.1016/j.future.2019.05.062.

[22] S. Alowayyed, T. Piontek, J. L. Suter, O. Hoenen, D. Groen, O. Luk, B. Bosak, P. Kopta, K. Kurowski, O. Perks, et al. Patterns for high performance multiscale computing. *Future Generation Computer Systems*, 91:335–346, 2019. doi:10.1016/j.future.2018.08.045.

[23] S. Alowayyed, D. Groen, P. V. Coveney, and A. G. Hoekstra. Multiscale computing in the exascale era. *Journal of Computational Science*, 22:15–25, 2017. doi:10.1016/j.jocs.2017.07.004.

[24] F. T. Bergmann, J. Cooper, M. König, I. Moraru, D. Nickerson, N. L. Novère, B. G. Olivier, S. Sahle, L. Smith, and D. Waltemath. Simulation experiment description markup language (SED-ML) level 1 version 3 (L1V3). *Journal of Integrative Bioinformatics*, 15(1):20170086, 2018. doi:doi:10.1515/jib-2017-0086.

[25] T. Warnke, T. Helms, and A. M. Uhrmacher. Reproducible and flexible simulation experiments with ML-Rules and SESSL. *Bioinformatics*, 34(8):1424–1427, 2017. doi:10.1093/bioinformatics/btx741.

[26] O. Dayıbaş, H. Oğuztüzün, and L. Yılmaz. On the use of model-driven engineering principles for the management of simulation experiments. *Journal of Simulation*, 13(2):83–95, 2019. doi:10.1080/17477778.2017.1418638.

[27] S. Robinson. Conceptual modeling for simulation. In *2013 Winter Simulations Conference (WSC)*, pages 377–388. IEEE, 2013. doi:10.1109/WSC.2013.6721435.

[28] A. Ruscheinski and A. M. Uhrmacher. Provenance in modeling and simulation studies — bridging gaps. In *2017 Winter Simulation Conference (WSC)*, pages 872–883. IEEE, 2017. doi:10.1109/WSC.2017.8247839.

[29] R. Nusse. Wnt signaling and stem cell control. *Cell Research*, 18(5):523–527, 2008. doi:10.1038/cr.2008.47.

[30] H. Clevers and R. Nusse. Wnt/β-catenin signaling and disease. *Cell*, 149(6):1192–1205, 2012. doi:10.1016/j.cell.2012.05.012.

[31] F. Haack, H. Lemcke, R. Ewald, T. Rharass, and A. M. Uhrmacher. Spatio-temporal model of endogenous ROS and raft-dependent WNT/beta-catenin signaling driving cell fate commitment in human neural progenitor cells. *PLOS Computational Biology*, 11(3), 2015. doi:10.1371/journal.pcbi.1004106.

[32] E. Lee, A. Salic, R. Krüger, R. Heinrich, and M. W. Kirschner. The roles of apc and axin derived from experimental and theoretical analysis of the wnt pathway. *PLOS Biology*, 1 (1), 2003. doi:10.1371/journal.pbio.0000010.

[33] B. jian Lin, S. hao Tsao, A. Chen, S.-K. Hu, L. Chao, and P. hsiu Grace Chao. Lipid rafts sense and direct electric field-induced migration. *Proceedings of the National Academy of Sciences*, 114(32):8568–8573, 2017. doi:10.1073/pnas.1702526114.

[34] F. Haack, K. Budde, and A. M. Uhrmacher. Exploring mechanistic and temporal regulation of LRP6 endocytosis in canonical WNT signaling. *Journal of Cell Science*, 133(15), 2020. doi:10.1242/jcs.243675.

[35] C. Maus, S. Rybacki, and A. M. Uhrmacher. Rule-based multi-level modeling of cell biological systems. *BMC Systems Biology*, 5(1):166, 2011. doi:10.1186/1752-0509-5-166.

[36] R. Ewald and A. M. Uhrmacher. SESSL: a domain-specific language for simulation experiments. *ACM Transactions on Modeling and Computer Simulation*, 24(2):11:1–11:25, 2014. doi:10.1145/2567895.

[37] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977. doi:10.1021/j100540a008.

[38] A. M. Law, W. D. Kelton, and W. D. Kelton. *Simulation modeling and analysis*, volume 3. McGraw-Hill New York, 2007.

[39] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation*. Pearson International, 5th edition, 2013. ISBN 9781292024370.

[40] T. Helms, T. Warnke, C. Maus, and A. M. Uhrmacher. Semantics and efficient simulation algorithms of an expressive multilevel modeling language. *ACM Transactions on Modeling and Computer Simulation*, 27(2), 2017. doi:10.1145/2998499.

[41] O. Mazemondet, M. John, S. Leye, A. Rolfs, and A. M. Uhrmacher. Elucidating the sources of β-catenin dynamics in human neural progenitor cells. *PLOS ONE*, 7(8):e42792–e42792, 2012. doi:10.1371/journal.pone.0042792.

[42] R. S. Malik-Sheriff, M. Glont, T. V. N. Nguyen, K. Tiwari, M. G. Roberts, A. Xavier, M. T. Vu, J. Men, M. Maire, S. Kananathan, E. L. Fairbanks, J. P. Meyer, et al. BioModels — 15 years of sharing computational models in life science. *Nucleic Acids Research*, 48(D1): D407–D415, 2020. doi:10.1093/nar/gkz1055.

[43] K. Budde, J. Smith, P. Wilsdorf, F. Haack, and A. M. Uhrmacher. Relating simulation studies by provenance—developing a family of Wnt signaling models. *PLOS Computational Biology*, 17(8):1–27, 2021. doi:10.1371/journal.pcbi.1009227.

[44] B. Lloyd-Lewis, A. G. Fletcher, T. C. Dale, and H. M. Byrne. Toward a quantitative understanding of the Wnt/β-catenin pathway through simulation and experiment. *WIREs Systems Biology and Medicine*, 5(4):391–407, 2013. doi:10.1002/wsbm.1221.

[45] A. L. MacLean, Z. Rosen, H. M. Byrne, and H. A. Harrington. Parameter-free methods distinguish Wnt pathway models and guide design of experiments. *Proceedings of the National Academy of Sciences*, 112(9):2652–2657, 2015. doi:10.1073/pnas.1416655112.

[46] J. Heller, C. Niemann, F. Plocksties, C. Haubelt, and D. Timmermann. Towards virtual prototyping of electrically active implants using systemc-ams. In *MBMV 2020 - Methods and Description Languages for Modelling and Verification of Circuits and Systems; GMM/ITG/GI-Workshop*, pages 1–8. VDE, 2020. ISBN 978-3-8007-5220-1.

[47] M. Jakobs, A. Fomenko, A. M. Lozano, and K. L. Kiening. Cellular, molecular, and clinical mechanisms of action of deep brain stimulation—a systematic review on established indications and outlook on future developments. *EMBO Molecular Medicine*, 11(4):e9575, 2019. doi:10.15252/emmm.201809575.

[48] C. Baunez and P. Gubellini. Chapter 12 - Effects of GPi and STN inactivation on physiological, motor, cognitive and motivational processes in animal models of Parkinson's disease. In A. Björklund and M. A. Cenci, editors, *Recent Advances in Parkinson's Disease: Basic Research*, volume 183 of *Progress in Brain Research*, pages 235–258. Elsevier, 2010. doi:10.1016/S0079-6123(10)83012-2.

[49] F. Plocksties, M. Kober, C. Niemann, J. Heller, M. Fauser, M. Nüssel, F. Uster, D. Franz, M. Zwar, A. Lüttig, J. Kröger, J. Harloff, A. Schulz, A. Richter, R. Köhling, D. Timmermann, and A. Storch. The software defined implantable modular platform (STELLA) for preclinical deep brain stimulation research in rodents. *Journal of Neural Engineering*, 18(5):056032, 2021. doi:10.1088/1741-2552/ac23e1.

[50] A. Vachoux, C. Grimm, and K. Einwich. Analog and mixed signal modelling with systemc-ams. In *2003 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 3, pages III–III, 2003. doi:10.1109/ISCAS.2003.1205169.

[51] B. P. Zeigler, A. Muzy, and E. Kofman. *Theory of modeling and simulation: discrete event & iterative system computational foundations*. Academic Press, 2018.

[52] A. Ruscheinski, P. Wilsdorf, J. Zimmermann, U. van Rienen, and A. M. Uhrmacher. An artefact-based workflow for finite element simulation studies. *Simulation Modelling Practice and Theory*, 116:102464, 2022. doi:10.1016/j.simpat.2021.102464.

[53] L. P. da Silva, S. C. Kundu, R. L. Reis, and V. M. Correlo. Electric phenomenon: A disregarded tool in tissue engineering and regenerative medicine. *Trends in Biotechnology*, 38(1):24–49, 2020. doi:10.1016/j.tibtech.2019.07.002.

[54] J. K. Krauss, N. Lipsman, T. Aziz, A. Boutet, P. Brown, J. W. Chang, B. Davidson, W. M. Grill, M. I. Hariz, A. Horn, et al. Technology of deep brain stimulation: current status and future directions. *Nature Reviews Neurology*, 17(2):75–87, 2021. doi:10.1038/s41582-020-00426-z.

[55] S. Meng, M. Rouabhia, and Z. Zhang. Electrical stimulation modulates osteoblast proliferation and bone protein production through heparin-bioactivated conductive scaffolds. *Bioelectromagnetics*, 34(3):189–199, 2013. doi:10.1002/bem.21766.

[56] J. Zimmermann, K. Budde, N. Arbeiter, F. Molina, A. Storch, A. M. Uhrmacher, and U. van Rienen. Using a digital twin of an electrical stimulation device to monitor and control the electrical stimulation of cells in vitro. *Frontiers in Bioengineering and Biotechnology*, 9, 2021. doi:10.3389/fbioe.2021.765516.

[57] M. Griffin, S. A. Iqbal, A. Sebastian, J. Colthurst, and A. Bayat. Degenerate wave and capacitive coupling increase human MSC invasion and proliferation while reducing cytotoxicity in an in vitro wound healing model. *PLOS One*, 6(8), 2011. doi:10.1371/journal.pone.0023404.

[58] K. Budde, J. Zimmermann, E. Neuhaus, M. Schröder, A. M. Uhrmacher, and U. van Rienen. Requirements for documenting electrical cell stimulation experiments for replicability and numerical modeling. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1082–1088, 2019. doi:10.1109/EMBC.2019.8856863.

[59] J. Zimmermann. EMStimTools, 2020. URL `https://github.com/j-zimmermann/EMStimTools/`. Last accessed 19 July, 2024.

[60] A. Logg, K.-A. Mardall, and G. N. Wells. *Automated Solution of Differential Equations by the Finite Element Method. The FEniCS Book.* Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-23099-8.

[61] P. Morin, R. H. Nochetto, and K. G. Siebert. Convergence of adaptive finite element methods. *SIAM Review*, 44(4):631–658, 2002. doi:10.1137/S0036144502409093.

[62] A. Erdemir, T. M. Guess, J. Halloran, S. C. Tadepalli, and T. M. Morrison. Considerations for reporting finite element analysis studies in biomechanics. *Journal of Biomechanics*, 45 (4):625–633, 2012. doi:10.1016/j.jbiomech.2011.11.038.

[63] O. Reinhardt, A. M. Uhrmacher, M. Hinsch, and J. Bijak. Developing agent-based migration models in pairs. In *2019 Winter Simulation Conference (WSC)*, pages 2713–2724, 2019. doi:10.1109/WSC40007.2019.9004946.

[64] S. Castles, H. De Haas, and M. J. Miller. *The age of migration: International population movements in the modern world.* Guilford Press, 5th edition, 2014. ISBN 9781462513116.

[65] J. Bijak. *Towards Bayesian Model-Based Demography: Agency, Complexity and Uncertainty in Migration Studies.* Springer Cham, 2022. doi:10.1007/978-3-030-83039-7.

[66] M. Hinsch and J. Bijak. The Effects of Information on the Formation of Migration Routes and the Dynamics of Migration. *Artificial Life*, 29(1):3–20, 2023. doi:10.1162/artl_a_00388.

[67] M. Hinsch and J. Bijak. *Principles and State of the Art of Agent-Based Migration Modelling*, pages 33–49. Springer International Publishing, Cham, 2022. doi:10.1007/978-3-030-83039-7_3.

[68] M. Hinsch and J. Bijak. Rumours lead to self-organized migration routes. In *ABMHuB 2019 International Workshop on Agent-Based Modelling of Human Behaviour*, 2019. URL `http://abmhub.cs.ucl.ac.uk/2019/`. Last accessed 19 July, 2024.

[69] M. Hinsch and J. Bijak. Towards more realistic models. In *Towards Bayesian Model-Based Demography: Agency, Complexity and Uncertainty in Migration Studies*, volume 17 of *Methodos Series*, chapter 8. Springer, Dordecht, 2022. doi:10.1007/978-3-030-83039-7_8.

[70] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: a fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. doi:10.1137/141000671.

[71] O. Reinhardt, T. Warnke, and A. M. Uhrmacher. A language for agent-based discrete-event modeling and simulation of linked lives. *ACM Transactions on Modeling and Computer Simulation*, 32(1), 2022. doi:10.1145/3486634.

[72] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022. URL `https://www.R-project.org/`. Last accessed 19 July, 2024.

[73] O. Reinhardt, T. Warnke, and A. M. Uhrmacher. Agent-based modelling and simulation with domain-specific languages. In *Towards Bayesian Model-Based Demography: Agency, Complexity and Uncertainty in Migration Studies*, volume 17 of *Methodos Series*, chapter 7. Springer, Dordecht, 2022. doi:10.1007/978-3-030-83039-7_7.

[74] A. Maria. Introduction to Modeling and Simulation. In *Proceedings of the 29th Conference on Winter Simulation*, WSC '97, pages 7–13, Washington, DC, USA, 1997. IEEE Computer Society. doi:10.1145/268437.268440.

[75] V. Grimm, S. F. Railsback, C. E. Vincenot, U. Berger, C. Gallagher, D. L. DeAngelis, B. Edmonds, J. Ge, J. Giske, J. Groeneveld, et al. The ODD protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism. *Journal of Artificial Societies and Social Simulation*, 23(2), 2020. doi:10.18564/jasss.4259.

[76] J. P. C. Kleijnen, S. M. Sanchez, T. W. Lucas, and T. M. Cioppa. State-of-the-art review: A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, 17(3):263–289, 2005. doi:10.1287/ijoc.1050.0136.

[77] R. G. Jaeger and T. R. Halliday. On confirmatory versus exploratory research. *Herpetologica*, 54:S64–S66, 1998. ISSN 0018-0831.

[78] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation*. Prentice Hall, 2nd edition, 1996. ISBN 9780132182317.

[79] I. J. Timm and F. Lorig. A survey on methodological aspects of computer simulation as research technique. In *2015 Winter Simulation Conference (WSC)*, pages 2704–2715, 2015. doi:10.1109/WSC.2015.7408377.

[80] O. Balci. Validation, verification, and testing techniques throughout the life cycle of a simulation study. In *Proceedings of Winter Simulation Conference*, pages 215–220, 1994. doi:10.1109/WSC.1994.717129.

[81] J. P. C. Kleijnen and R. G. Sargent. A methodology for fitting and validating metamodels in simulation. *European Journal of Operational Research*, 120(1):14–29, 2000. doi:10.1016/S0377-2217(98)00392-0.

[82] S. Robinson. Conceptual modelling for simulation part I: definition and requirements. *Journal of the Operational Research Society*, 59(3):278–290, 2008. doi:10.1057/palgrave.jors.2602368.

[83] P. Wilsdorf, F. Haack, and A. M. Uhrmacher. Conceptual models in simulation studies: Making it explicit. In *2020 Winter Simulation Conference (WSC)*, pages 2353–2364. IEEE, 2020. doi:10.1109/WSC48552.2020.9383984.

[84] J. Liu, Y. Yu, L. Zhang, and C. Nie. An overview of conceptual model for simulation and its validation. *Procedia Engineering*, 24:152–158, 2011. doi:10.1016/j.proeng.2011.11.2618.

[85] J. P. C. Kleijnen. Verification and validation of simulation models. *European Journal of Operational Research*, 82(1):145 – 162, 1995. doi:10.1016/0377-2217(94)00016-6.

[86] J. P. C. Kleijnen. Design of experiments: Overview. In *2008 Winter Simulation Conference*, pages 479–488. IEEE, 2008. doi:10.1109/WSC.2008.4736103.

[87] B. Iooss, L. Boussouf, V. Feuillard, and A. Marrel. Numerical studies of the metamodel fitting and validation processes. *arXiv*, 2010. doi:10.48550/arXiv.1001.1049. Preprint.

[88] T. E. Turner, S. Schnell, and K. Burrage. Stochastic approaches for modelling in vivo reactions. *Computational Biology and Chemistry*, 28(3):165–178, 2004. doi:10.1016/j.compbiolchem.2004.05.001.

[89] M. E. Johnson, A. Chen, J. R. Faeder, P. Henning, I. I. Moraru, M. Meier-Schellersheim, R. F. Murphy, T. Prüstel, J. A. Theriot, and A. M. Uhrmacher. Quantifying the roles of space and stochasticity in computer simulations for cell biology and cellular biochemistry. *Molecular Biology of the Cell*, 32(2):186–210, 2021. doi:10.1091/mbc.E20-08-0530.

[90] K. Burrage, P. M. Burrage, A. Leier, T. Marquez-Lago, and D. V. Nicolau. Stochastic simulation for spatial modelling of dynamic processes in a living cell. In H. Koeppl, G. Setti, M. di Bernardo, and D. Densmore, editors, *Design and Analysis of Biomolecular Circuits*, pages 43–62. Springer, 2011. doi:10.1007/978-1-4419-6766-4_2.

[91] F. Haack, K. Burrage, R. Redmer, and A. M. Uhrmacher. Studying the role of lipid rafts on protein receptor bindings with cellular automata. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(3):760–770, 2013. doi:10.1109/TCBB.2013.40.

[92] K. Burrage and P. M. Burrage. High strong order explicit Runge-Kutta methods for stochastic ordinary differential equations. *Applied Numerical Mathematics*, 22(1-3):81–101, 1996. doi:10.1016/S0168-9274(96)00027-X.

[93] T. Tian and K. Burrage. Binomial leap methods for simulating stochastic chemical kinetics. *The Journal of Chemical Physics*, 121(21):10356–10364, 2004. doi:10.1063/1.1810475.

[94] M. John, C. Lhoussaine, J. Niehren, and A. M. Uhrmacher. The attributed pi-calculus with priorities. In *Transactions on Computational Systems Biology XII*, Lecture Notes in Computer Science, pages 13–76. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-11712-1_2.

[95] B. A. J. Lawson, C. C. Drovandi, N. Cusimano, P. Burrage, B. Rodriguez, and K. Burrage. Unlocking data sets by calibrating populations of models to data density: A study in atrial electrophysiology. *Science Advances*, 4(1), 2018. doi:10.1126/sciadv.1701676.

[96] J. Hasenauer, N. Jagiella, S. Hross, and F. J. Theis. Data-driven modelling of biological multi-scale processes. *Journal of Coupled Systems and Multiscale Dynamics*, 3(2):101–121, 2015. doi:10.1166/jcsmd.2015.1069.

[97] P. Wilsdorf, M. Zuska, P. Andelfinger, A. M. Uhrmacher, and F. Peters. Validation without data - formalizing stylized facts of time series. In *2023 Winter Simulation Conference (WSC)*, pages 2674–2685, 2023. doi:10.1109/WSC60868.2023.10408388.

[98] S. Singh, M. Weeber, and K.-P. Birke. Advancing digital twin implementation: a toolbox for modelling and simulation. *Procedia CIRP*, 99:567–572, 2021. doi:10.1016/j.procir.2021.03.078.

[99] H. Aydt, S. J. Turner, W. Cai, and M. Y. H. Low. Research issues in symbiotic simulation. In *2009 Winter Simulation Conference (WSC)*, pages 1213–1222. IEEE, 2009. doi:10.1109/WSC.2009.5429419.

[100] O. Reinhardt, T. Warnke, A. Ruscheinski, and A. M. Uhrmacher. Valid and reproducible simulation studies—making it explicit. In C. Beisbart and N. J. Saam, editors, *Computer Simulation Validation. Fundamental Concepts, Methodological Frameworks, Philosophical Perspectives*, pages 607–672. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-319-70766-2_25.

[101] O. Balci. Principles and techniques of simulation validation, verification, and testing. In *Proceedings of the 27th conference on Winter simulation*, WSC '95, pages 147–154. IEEE Computer Society, 1995. doi:10.1145/224401.224456.

[102] R. R. Barton. Designing simulation experiments. In *2013 Winter Simulation Conference (WSC), 2013*. IEEE, 2013. doi:10.1109/WSC.2013.6721432.

[103] S. Razavi, A. Jakeman, A. Saltelli, C. Prieur, B. Iooss, E. Borgonovo, E. Plischke, S. Lo Piano, T. Iwanaga, W. Becker, S. Tarantola, J. H. A. Guillaume, et al. The future of sensitivity analysis: An essential discipline for systems modeling and policy support. *Environmental Modelling & Software*, 137:104954, 2021. doi:10.1016/j.envsoft.2020.104954.

[104] D. Peng. *Reusing Simulation Experiments for Model Composition and Extension*. PhD thesis, University of Rostock, 2017. URL `https://rosdok.uni-rostock.de/resolve/id/rosdok_disshab_0000001686`. Last accessed 19 July, 2024.

[105] K. Pawlikowski. Steady-state simulation of queueing processes: Survey of problems and solutions. *ACM Computing Surveys*, 22(2):123–170, 1990. doi:10.1145/78919.78921.

[106] J. Júlvez. A simulation approach to detect oscillating behaviour in stochastic population models. In *Computational Methods in Systems Biology: 11th International Conference, CMSB 2013*, volume 8130, page 273. Springer, 2013.

[107] T. Eißing, S. Waldherr, F. Allgöwer, P. Scheurich, and E. Bullinger. Steady state and (bi-) stability evaluation of simple protease signalling networks. *Biosystems*, 90(3):591–601, 2007. doi:10.1016/j.biosystems.2007.01.003.

[108] N. Sri Namachchivaya. Stochastic bifurcation. *Applied Mathematics and Computation*, 38 (2):101–159, 1990. doi:10.1016/0096-3003(90)90051-4.

[109] G. Iooss and D. D. Joseph. *Elementary stability and bifurcation theory*. Springer New York, NY, 2012. doi:10.1007/978-1-4684-9336-8.

[110] S. Leye, R. Ewald, and A. M. Uhrmacher. Composing problem solvers for simulation experimentation: A case study on steady state estimation. *PLOS ONE*, 9(4):1–13, 2014. doi:10.1371/journal.pone.0091948.

[111] K. White, M. Cobb, and S. Spratt. A comparison of five steady-state truncation heuristics for simulation. In *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, volume 1, pages 755–760 vol.1, 2000. doi:10.1109/WSC.2000.899843.

[112] C. Alexopoulos and D. Goldsman. To batch or not to batch? *ACM Transactions on Modeling and Computer Simulation*, 14(1):76–114, 2004. doi:10.1145/974734.974738.

[113] S. Robinson. Automated analysis of simulation output data. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 8 pp.–, 2005. doi:10.1109/WSC.2005.1574320.

[114] M. Kim, S. H. Yoon, P. A. Domanski, and W. Vance Payne. Design of a steady-state detector for fault detection and diagnosis of a residential air conditioner. *International Journal of Refrigeration*, 31(5):790–799, 2008. doi:10.1016/j.ijrefrig.2007.11.008.

[115] D. G. Cacuci, M. Ionescu-Bujor, and I. M. Navon. *Sensitivity and Uncertainty Analysis, Volume II: Applications to Large-Scale Systems*. CRC press, 2005. doi:10.1201/9780203483572.

[116] F. Campolongo, A. Saltelli, and J. Cariboni. From screening to quantitative sensitivity analysis. A unified approach. *Computer Physics Communications*, 182(4):978–988, 2011. doi:10.1016/j.cpc.2010.12.039.

[117] W. Edeling, H. Arabnejad, R. Sinclair, D. Suleimenova, K. Gopalakrishnan, B. Bosak, D. Groen, I. Mahmood, D. Crommelin, and P. V. Coveney. The impact of uncertainty on predictions of the CovidSim epidemiological code. *Nature Computational Science*, 1(2): 128–135, 2021. doi:10.1038/s43588-021-00028-9.

[118] J. P. C. Kleijnen. Experimental design for sensitivity analysis, optimization, and validation of simulation models. In J. Banks, editor, *Handbook of Simulation*, pages 173–223. John Wiley & Sons, Inc, New York, 1998. doi:10.1002/9780470172445.ch6.

[119] D. Goldsman and B. L. Nelson. Ranking, selection, and multiple comparisons in computer simulation. In J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, editors, *Proceedings of the Winter Simulation Conference*, pages 192–199. IEEE, 1994. doi:10.1109/WSC.1994.717120.

[120] S. M. Sanchez. Work smarter, not harder: Guidelines for designing simulation experiments. In *Proceedings of the 2005 Winter Simulation Conference*, pages 14–14. IEEE, 2005. doi:10.1109/WSC.2005.1574241.

[121] N. A. Butler. Optimal and orthogonal Latin hypercube designs for computer experiments. *Biometrika*, 88(3):847–857, 10 2001. doi:10.1093/biomet/88.3.847.

[122] C. D. Lin, R. Mukerjee, and B. Tang. Construction of orthogonal and nearly orthogonal Latin hypercubes. *Biometrika*, 96(1):243–247, 2009. doi:10.1093/biomet/asn064.

[123] T. M. Cioppa and T. W. Lucas. Efficient Nearly Orthogonal and Space-Filling Latin Hypercubes. *Technometrics*, 49(1):45–55, 2007. doi:10.1198/004017006000000453.

[124] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. John Wiley & Sons, 2004. ISBN 0-470-87093-1.

[125] F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B. Stephenson, and T. Wagener. Sensitivity analysis of environmental models: A systematic review with practical workflow. *Environmental Modelling & Software*, 79:214–232, 2016. doi:10.1016/j.envsoft.2016.02.008.

[126] A. Saltelli, K. Aleksankina, W. Becker, P. Fennell, F. Ferretti, N. Holst, S. Li, and Q. Wu. Why so many published sensitivity analyses are false: A systematic review of sensitivity analysis practices. *Environmental Modelling & Software*, 114:29–39, 2019. doi:10.1016/j.envsoft.2019.01.012.

[127] E. Borgonovo and E. Plischke. Sensitivity analysis: A review of recent advances. *European Journal of Operational Research*, 248(3):869–887, 2016. doi:10.1016/j.ejor.2015.06.032.

[128] A. Saltelli. Sensitivity Analysis for Importance Assessment. *Risk Analysis*, 22(3):579–590, 2002. doi:10.1111/0272-4332.00040.

[129] D. Suleimenova, H. Arabnejad, W. N. Edeling, and D. Groen. Sensitivity-driven simulation development: A case study in forced migration. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2197):20200077, 2021. doi:10.1098/rsta.2020.0077.

[130] Y.-C. Ho. Performance evaluation and perturbation analysis of discrete event dynamic systems. *IEEE Transactions on Automatic Control*, 32(7):563–572, 1987. doi:10.1109/TAC.1987.1104665.

[131] M. D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991. doi:10.1080/00401706.1991.10484804.

[132] I. M. Sobol. Sensitivity estimates for nonlinear mathematical models. *Mathematical Modelling and Computational Experiments*, 1(4):407–414, 1993.

[133] R. Cukier, C. Fortuin, K. Shuler, A. Petschek, and J. Schaibly. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. I theory. *Journal of Chemical Physics*, 59(8):3873 – 3878, 1973. doi:10.1063/1.1680571.

[134] J. B. Nagel. Bayesian techniques for inverse uncertainty quantification. *IBK Bericht*, 504, 2019. URL `https://ethz.ch/content/dam/ethz/special-interest/baug/ibk/risk-safety-and-uncertainty-dam/publications/doctoral-theses/ThesisNagel.pdf`. PhD thesis, ETH Zürich, last accessed 19 July, 2024.

[135] A. P. Galvão Scheidegger, A. Banerjee, and T. F. Pereira. Uncertainty quantification in simulation models: A proposed framework and application through case study. In *2018 Winter Simulation Conference (WSC)*, pages 1599–1610. IEEE, 2018. doi:10.1109/WSC.2018.8632281.

[136] M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001. doi:10.1111/1467-9868.00294.

[137] S. Reinker, R. M. Altman, and J. Timmer. Parameter estimation in stochastic biochemical reactions. *IEE Proceedings - Systems Biology*, 153(4):168–178, 2006. doi:10.1049/ip-syb:20050105.

[138] E. D. Mitra, R. Suderman, J. Colvin, A. Ionkov, A. Hu, H. M. Sauro, R. G. Posner, and W. S. Hlavacek. PyBioNetFit and the biological property specification language. *iScience*, 19:1012–1036, 2019. doi:10.1016/j.isci.2019.08.045.

[139] K. Cranmer, J. Brehmer, and G. Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020. doi:10.1073/pnas.1912789117.

[140] B. M. Turner and T. V. Zandt. A tutorial on approximate Bayesian computation. *Journal of Mathematical Psychology*, 56(2):69–85, 2012. doi:10.1016/j.jmp.2012.02.005.

[141] M. Sunnåker, A. G. Busetto, E. Numminen, J. Corander, M. Foll, and C. Dessimoz. Approximate bayesian computation. *PLOS Computational Biology*, 9(1):e1002803, 2013. doi:10.1371/journal.pcbi.1002803.

[142] J. Lintusaari, M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander. Fundamentals and recent developments in approximate bayesian computation. *Systematic Biology*, 66(1): e66–e82, 2017. doi:10.1093/sysbio/syw077.

[143] R. Dutta, M. Schoengens, J.-P. Onnela, and A. Mira. ABCpy: A user-friendly, extensible, and parallel library for approximate bayesian computation. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '17, New York, NY, USA, 2017. ACM. doi:10.1145/3093172.3093233.

[144] T. O. Hodson. Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not. *Geoscientific Model Development*, 15(14):5481–5487, 2022. doi:10.5194/gmd-15-5481-2022.

[145] P. J. Gonçalves, J.-M. Lueckmann, M. Deistler, M. Nonnenmacher, K. Öcal, G. Bassetto, C. Chintaluri, W. F. Podlaski, S. A. Haddad, T. P. Vogels, D. S. Greenberg, and J. H. Macke. Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*, 9:e56261, 2020. doi:10.7554/eLife.56261.

[146] S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380, 2016. doi:10.1007/s10479-015-2019-x.

[147] A. M. Law and M. G. McComas. Simulation-based optimization. In *Proceedings of the Winter Simulation Conference*, volume 1, pages 41–44. IEEE, 2002. doi:10.1109/WSC.2002.1172866.

[148] M. C. Fu. Feature Article: Optimization for simulation: Theory vs. Practice. *INFORMS Journal on Computing*, 14(3):192–215, 2002. doi:10.1287/ijoc.14.3.192.113.

[149] A.-T. Nguyen, S. Reiter, and P. Rigo. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, 113:1043–1058, 2014. doi:10.1016/j.apenergy.2013.08.061.

[150] O. Reinhardt, J. Hilton, T. Warnke, J. Bijak, and A. M. Uhrmacher. Streamlining simulation experiments with agent-based models in demography. *Journal of Artificial Societies and Social Simulation*, 21(3):9, 2018. doi:10.18564/jasss.3784.

[151] T. Trucano, L. Swiler, T. Igusa, W. Oberkampf, and M. Pilch. Calibration, validation, and sensitivity analysis: what's what. *Reliability Engineering & System Safety*, 91(10): 1331–1357, 2006. doi:10.1016/j.ress.2005.11.031.

[152] M. S. Meketon. Optimization in simulation: A survey of recent results. In *Proceedings of the 19th Conference on Winter Simulation*, WSC '87, pages 58–67, Atlanta, Georgia, USA, 1987. ACM. doi:10.1145/318371.318384.

[153] Y. Carson and A. Maria. Simulation optimization: Methods and applications. In *Proceedings of the 29th Conference on Winter Simulation*, WSC '97, page 118–126. IEEE Computer Society, 1997. doi:10.1145/268437.268460.

[154] H.-G. Beyer and B. Sendhoff. Robust optimization – a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33):3190–3218, 2007. doi:10.1016/j.cma.2007.03.003.

[155] G. Figueira and B. Almada-Lobo. Hybrid simulation–optimization methods: A taxonomy and discussion. *Simulation Modelling Practice and Theory*, 46:118–134, 2014. doi:10.1016/j.simpat.2014.03.007.

[156] A. F. Villaverde, F. Fröhlich, D. Weindl, J. Hasenauer, and J. R. Banga. Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinformatics*, 35 (5):830–838, 2018. doi:10.1093/bioinformatics/bty736.

[157] S. M. Sanchez and P. J. Sanchez. Robustness revisited: Simulation optimization viewed through a different lens. In *2020 Winter Simulation Conference (WSC)*, pages 60–74, 2020. doi:10.1109/WSC48552.2020.9383880.

[158] E. Hassannayebi, A. Sajedinejad, and S. Mardani. Urban rail transit planning using a two-stage simulation-based optimization approach. *Simulation Modelling Practice and Theory*, 49:151–166, 2014. doi:10.1016/j.simpat.2014.09.004.

[159] G. Agha and K. Palmskog. A survey of statistical model checking. *ACM Transactions on Modeling and Computer Simulation*, 28(1):1–39, 2018. doi:10.1145/3158668.

[160] S. Gilmore, D. Reijsbergen, and A. Vandin. Transient and Steady-State Statistical Analysis for Discrete Event Simulators. In N. Polikarpova and S. Schneider, editors, *Integrated Formal Methods*, Lecture Notes in Computer Science, pages 145–160. Springer Cham, 2017. doi:10.1007/978-3-319-66845-1_10.

[161] M. AlTurki and J. Meseguer. PVeStA: A parallel statistical model checking and quantitative analysis tool. In *Algebra and Coalgebra in Computer Science*, Lecture Notes in Computer Science, pages 386–392. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-22944-2_28.

[162] F. Hussain, C. J. Langmead, Q. Mi, J. Dutta-Moscato, Y. Vodovotz, and S. K. Jha. Automated parameter estimation for biological models using bayesian statistical model checking. *BMC Bioinformatics*, 16(17):1–14, 2015. doi:10.1186/1471-2105-16-S17-S8.

[163] L. Bortolussi, D. Milios, and G. Sanguinetti. Smoothed model checking for uncertain continuous-time Markov chains. *Information and Computation*, 247:235–253, 2016. doi:10.1016/j.ic.2016.01.004.

[164] K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification*, Lecture Notes in Computer Science, pages 266–280. Springer Berlin Heidelberg, 2005. doi:10.1007/11513988_26.

[165] H. L. S. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer*, 8(3):216–228, 2006. doi:10.1007/s10009-005-0187-8.

[166] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945. doi:10.1214/aoms/1177731118.

[167] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004. doi:10.1007/978-3-540-30206-3_12.

[168] A. Donzé, T. Ferrere, and O. Maler. Efficient robust monitoring for STL. In *International Conference on Computer Aided Verification*, Lecture Notes in Computer Science, pages 264–279. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-39799-8_19.

[169] N. Kamide. Bounded linear-time temporal logic: A proof-theoretic investigation. *Annals of Pure and Applied Logic*, 163(4):439–466, 2012. doi:10.1016/j.apal.2011.12.002.

[170] L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink. Qualitative and quantitative monitoring of spatio-temporal properties with SSTL. *Logical Methods in Computer Science*, 14(4), 2018. doi:10.23638/LMCS-14(4:2)2018.

[171] S. Sebastio and A. Vandin. MultiVeStA: Statistical model checking for discrete event simulators. Working Paper, IMT Institute for Advanced Studies Lucca, Turin, Italy, 2013. URL `http://eprints.imtlucca.it/1798/`. Last accessed 19 July, 2024.

[172] M. Golfarelli, S. Rizzi, and A. Proli. Designing what-if analysis: Towards a methodology. In *Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP*, DOLAP '06, pages 51–58, New York, NY, USA, 2006. AsCM. doi:10.1145/1183512.1183523.

[173] N. Ghaffarzadegan. Simulation-based what-if analysis for controlling the spread of Covid-19 in universities. *PLOS ONE*, 16(2):e0246323, 2021. doi:10.1371/journal.pone.0246323.

[174] S. Bankes. Exploratory modeling for policy analysis. *Operations research*, 41(3):435–449, 1993. doi:10.1287/opre.41.3.435.

[175] E. A. Moallemi, J. Kwakkel, F. J. de Haan, and B. A. Bryan. Exploratory modeling for analyzing coupled human-natural systems under uncertainty. *Global Environmental Change*, 65:102186, 2020. doi:10.1016/j.gloenvcha.2020.102186.

[176] J. Bijak, P. A. Higham, J. Hilton, M. Hinsch, S. Nurse, T. Prike, P. W. Smith, O. Reinhardt, and A. M. Uhrmacher. Modelling migration: Decisions, processes and outcomes. In *2020 Winter Simulation Conference (WSC)*, pages 2613–2624. IEEE, 2020. doi:10.1109/WSC48552.2020.9384072.

[177] M. Amer, T. U. Daim, and A. Jetter. A review of scenario planning. *Futures*, 46:23–40, 2013. doi:10.1016/j.futures.2012.10.003.

[178] Q. V. H. Nguyen, K. Zheng, M. Weidlich, B. Zheng, H. Yin, T. T. Nguyen, and B. Stantic. What-if analysis with conflicting goals: Recommending data ranges for exploration. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 89–100, 2018. doi:10.1109/ICDE.2018.00018.

[179] J. H. Kwakkel. The exploratory modeling workbench: An open source toolkit for exploratory modeling, scenario discovery, and (multi-objective) robust decision making. *Environmental Modelling & Software*, 96:239–250, 2017. doi:10.1016/j.envsoft.2017.06.054.

[180] F. Pires, B. Ahmad, A. P. Moreira, and P. Leitão. Digital twin based what-if simulation for energy management. In *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 309–314, 2021. doi:10.1109/ICPS49255.2021.9468224.

[181] W. Cui. Visual analytics: A comprehensive overview. *IEEE Access*, 7:81555–81573, 2019. doi:10.1109/ACCESS.2019.2923736.

[182] K. Gustafsson. Control of error and convergence in ODE solvers. 1992. URL `http://inis.iaea.org/Search/search.aspx?orig_q=RN:24008017`. Ph.D. thesis, Lund University, Sweden. Last accessed 19 July, 2024.

[183] S. Dallas, K. Machairas, and E. Papadopoulos. A comparison of ODE solvers for dynamical systems with impacts. *Journal of Computational and Nonlinear Dynamics, Special Issue on Dynamics of Systems with Impacts*, 2017. doi:10.1115/1.4037074.

[184] K. M. Kalayeh. Finite element convergence studies of a time-dependent test problem using COMSOL 5.1. 2015. doi:10.13016/M28P5VD4K. Technical Report.

[185] M. Bäker. How to get meaningful and correct results from your finite element model. *arXiv*, 2018. doi:10.48550/arXiv.1811.05753. Preprint.

[186] W. L. Oberkampf. Simulation accuracy, uncertainty, and predictive capability: A physical sciences perspective. In C. Beisbart and N. J. Saam, editors, *Computer Simulation Validation: Fundamental Concepts, Methodological Frameworks, and Philosophical Perspectives*, Simulation Foundations, Methods and Applications, pages 69–97. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-319-70766-2_3.

[187] I. Babuška and W. C. Rheinboldt. A-posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering*, 12(10):1597–1615, 1978. doi:10.1002/nme.1620121010.

[188] J.-F. Hiller and K.-J. Bathe. Measuring convergence of mixed finite element discretizations: an application to shell structures. *Computers & Structures*, 81(8):639–654, 2003. doi:10.1016/S0045-7949(03)00010-5.

[189] G. F. Riley and T. R. Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-12331-3_2.

[190] J. Himmelspach and A. M. Uhrmacher. Plug'n simulate. In *40th annual simulation symposium (ANSS'07)*, pages 137–143. IEEE, 2007. doi:10.1109/ANSS.2007.34.

[191] L. F. Perrone, C. S. Main, and B. C. Ward. SAFE: Simulation automation framework for experiments. In *2012 Winter Simulation Conference (WSC)*, pages 1–12, 2012. doi:10.1109/WSC.2012.6465286.

[192] J. Salecker, M. Sciaini, K. M. Meyer, and K. Wiegand. The NLRX R package: A next-generation framework for reproducible NetLogo model analyses. *Methods in Ecology and Evolution*, 10(11):1854–1863, 2019. doi:10.1111/2041-210X.13286.

[193] D. Waltemath, R. Adams, F. T. Bergmann, M. Hucka, F. Kolpakov, A. K. Miller, I. I. Moraru, D. Nickerson, S. Sahle, J. L. Snoep, et al. Reproducible computational biology experiments with SED-ML-the simulation experiment description markup language. *BMC Systems Biology*, 5(1):198, 2011. doi:10.1186/1752-0509-5-198.

[194] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. COPASI—a COmplex PAthway SImulator. *Bioinformatics*, 22(24): 3067–3074, 2006. doi:10.1093/bioinformatics/btl485.

[195] J. Cooper, M. Scharm, and G. R. Mirams. The cardiac electrophysiology web lab. *Biophysical journal*, 110(2):292–300, 2016. doi:10.1016/j.bpj.2015.12.012.

[196] A. C. Daly, M. Clerx, K. A. Beattie, J. Cooper, D. J. Gavaghan, and G. R. Mirams. Reproducible model development in the cardiac electrophysiology web lab. *Progress in Biophysics and Molecular Biology*, 139:3–14, 2018. doi:10.1016/j.pbiomolbio.2018.05.011.

[197] J. Schaber. Easy parameter identifiability analysis with copasi. *Biosystems*, 110(3):183–185, 2012. doi:10.1016/j.biosystems.2012.09.003.

[198] F. T. Bergmann, S. Hoops, B. Klahn, U. Kummer, P. Mendes, J. Pahle, and S. Sahle. Copasi and its applications in biotechnology. *Journal of Biotechnology*, 261:215–220, 2017. doi:10.1016/j.jbiotec.2017.06.1200. Bioinformatics Solutions for Big Data Analysis in Life Sciences presented by the German Network for Bioinformatics Infrastructure.

[199] J. Cooper, G. R. Mirams, and S. A. Niederer. High-throughput functional curation of cellular electrophysiology models. *Progress in Biophysics and Molecular Biology*, 107(1): 11–20, 2011. doi:10.1016/j.pbiomolbio.2011.06.003. Experimental and Computational Model Interactions in Bio-Research: State of the Art.

[200] P. Yordanov, J. Stelling, and I. Otero-Muras. BioSwitch: a tool for the detection of bistability and multi-steady state behaviour in signalling and gene regulatory networks. *Bioinformatics*, 36(5):1640–1641, 2020. doi:10.1093/bioinformatics/btz746.

[201] O. Enge-Rosenblatt, C. Clauß, P. Schneider, M. Vetter, and S. Schwunk. ComplexLib-a modelica library for steady-state analysis of ac circuits within phasor domain. In *7th International Modelica Conference*, 2009. ISBN 978-91-7393-513-5.

[202] M. Kennedy and G. Petropoulos. Chapter 17 - GEM-SA: The gaussian emulation machine for sensitivity analysis. In G. P. Petropoulos and P. K. Srivastava, editors, *Sensitivity Analysis in Earth Observation Modelling*, pages 341–361. Elsevier, 2017. doi:10.1016/B978-0-12-803011-0.00017-3.

[203] S. Tennøe, G. Halnes, and G. T. Einevoll. Uncertainpy: A python toolbox for uncertainty quantification and sensitivity analysis in computational neuroscience. *Frontiers in Neuroinformatics*, 12:49, 2018. doi:10.3389/fninf.2018.00049.

[204] R. Dutta, M. Schoengens, L. Pacchiardi, A. Ummadisingu, N. Widmer, P. Künzli, J.-P. Onnela, and A. Mira. ABCpy: A high-performance computing perspective to approximate bayesian computation. *Journal of Statistical Software*, 100(7):1–38, 2021. doi:10.18637/jss.v100.i07.

[205] A. Tejero-Cantero, J. Boelts, M. Deistler, J.-M. Lueckmann, C. Durkan, P. J. Gonçalves, D. S. Greenberg, and J. H. Macke. sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505, 2020. doi:10.21105/joss.02505.

[206] L. Schmiester, Y. Schälte, F. T. Bergmann, T. Camba, E. Dudkin, J. Egert, F. Fröhlich, L. Fuhrmann, A. L. Hauber, S. Kemmer, P. Lakrisenko, C. Loos, et al. PEtab—Interoperable specification of parameter estimation problems in systems biology. *PLOS Computational Biology*, 17(1):e1008646, 2021. doi:10.1371/journal.pcbi.1008646.

[207] R. Pasupathy and S. G. Henderson. SimOpt: A library of simulation optimization problems. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 4075–4085, 2011. doi:10.1109/WSC.2011.6148097.

[208] B. Boyer, K. Corre, A. Legay, and S. Sedwards. PLASMA-lab: A flexible, distributable statistical model checking library. In K. Joshi, M. Siegle, M. Stoelinga, and P. R. D'Argenio, editors, *Quantitative evaluation of systems*, pages 160–164. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40196-1.

[209] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In T. Field, P. G. Harrison, J. Bradley, and U. Harder, editors, *Computer Performance Evaluation: Modelling Techniques and Tools*, Lecture Notes in Computer Science, pages 200–204. Springer Berlin Heidelberg, 2002. doi:10.1007/3-540-46029-2_13.

[210] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker MRMC. *Performance Evaluation*, 68(2):90–104, 2011. doi:10.1016/j.peva.2010.04.001.

[211] D. Waltz and B. G. Buchanan. Automating science. *Science*, 324(5923):43–44, 2009. doi:10.1126/science.1172781.

[212] A. M. Uhrmacher, P. Frazier, R. Hähnle, F. Klügl, F. Lorig, B. Ludäscher, L. Nenzi, C. Ruiz-Martin, B. Rumpe, C. Szabo, G. Wainer, and P. Wilsdorf. Context, composition, automation, and communication - the C2AC roadmap for modeling and simulation. *ACM Transactions on Modeling and Computer Simulation*, 34(4), 2024. doi:10.1145/3673226.

[213] J. Xiao, P. Andelfinger, W. Cai, P. Richmond, A. Knoll, and D. Eckhoff. OpenABLext: An automatic code generation framework for agent-based simulations on CPU-GPU-FPGA heterogeneous platforms. *Concurrency and Computation: Practice and Experience*, 32(21): e5807, 2020. doi:10.1002/cpe.5807.

[214] M. Reiter, U. Breitenbucher, O. Kopp, and D. Karastoyanova. Quality of data driven simulation workflows. In *Conference on E-Science (e-Science)*. IEEE, 2012. doi:10.1109/eScience.2012.6404417.

[215] E. Deelman, T. Peterka, I. Altintas, C. D. Carothers, K. K. van Dam, K. Moreland, M. Parashar, L. Ramakrishnan, M. Taufer, and J. Vetter. The future of scientific workflows. *The International Journal of High Performance Computing Applications*, 32(1):159–175, 2018. doi:10.1177/1094342017704893.

[216] T. Helms, J. Himmelspach, C. Maus, O. Röwer, J. Schützel, and A. M. Uhrmacher. Toward a language for the flexible observation of simulations. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pages 1–12. IEEE, 2012. doi:10.1109/WSC.2012.6465073.

[217] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan. Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. In *Lectures on Runtime Verification*, Lecture Notes in Computer Science, pages 135–175. Springer, Cham, 2018. doi:10.1007/978-3-319-75632-5_5.

[218] L. Nenzi, E. Bartocci, L. Bortolussi, and M. Loreti. A logic for monitoring dynamic networks of spatially-distributed cyber-physical systems. *Logical Methods in Computer Science*, 18(1), 2022. doi:10.46298/lmcs-18(1:4)2022.

[219] F. Lorig. *Hypothesis-Driven Simulation Studies: Assistance for the Systematic Design and Conducting of Computer Simulation Experiments*. Springer, 2019. doi:10.1007/978-3-658-27588-4_5.

[220] A. Ruscheinski, K. Budde, T. Warnke, P. Wilsdorf, B. C. Hiller, M. Dombrowsky, and A. M. Uhrmacher. Generating simulation experiments based on model documentations and templates. In *2018 Winter Simulation Conference (WSC)*, pages 715–726. IEEE, 2018. doi:10.1109/WSC.2018.8632515.

[221] P. Wilsdorf, M. Dombrowsky, A. M. Uhrmacher, J. Zimmermann, and U. v. Rienen. Simulation experiment schemas – beyond tools and simulation approaches. In *2019 Winter Simulation Conference (WSC)*, pages 2783–2794. IEEE, 2019. doi:10.1109/WSC40007.2019.9004710.

[222] A. Teran-Somohano, A. E. Smith, J. Ledet, L. Yilmaz, and H. Oğuztüzün. A model-driven engineering approach to simulation experiment design and execution. In *2015 Winter Simulation Conference (WSC)*, pages 2632–2643, 2015. doi:10.1109/WSC.2015.7408371.

[223] A. Ruscheinski, T. Warnke, and A. M. Uhrmacher. Artifact-based workflows for supporting simulation studies. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1064–1078, 2019. doi:10.1109/TKDE.2019.2899840.

[224] T. Helms, R. Ewald, S. Rybacki, and A. M. Uhrmacher. Automatic Runtime Adaptation for Component-Based Simulation Algorithms. *ACM Transactions on Modeling and Computer Simulation*, 26(1):7:1–7:24, 2015. doi:10.1145/2821509.

[225] R. Ewald, R. Schulz, and A. M. Uhrmacher. Selecting simulation algorithm portfolios by genetic algorithms. In *2010 IEEE Workshop on Principles of Advanced and Distributed Simulation*, pages 1–9, 2010. doi:10.1109/PADS.2010.5471673.

[226] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA, 2019. ACM. doi:10.1145/3292500.3330701.

[227] L. Yilmaz, S. Chakladar, and K. Doud. The goal-hypothesis-experiment framework: a generative cognitive domain architecture for simulation experiment management. In *2016 Winter Simulation Conference (WSC)*, pages 1001–1012. IEEE, 2016. doi:10.1109/WSC.2016.7822160.

[228] D. Peng, T. Warnke, F. Haack, and A. M. Uhrmacher. Reusing simulation experiment specifications in developing models by successive composition—a case study of the Wnt/β-catenin signaling pathway. *SIMULATION*, 93(8):659–677, 2017. doi:10.1177/00375497177043.

[229] D. Peng, T. Warnke, F. Haack, and A. M. Uhrmacher. Reusing simulation experiment specifications to support developing models by successive extension. *Simulation Modelling Practice and Theory*, 68:33–53, 2016. doi:10.1016/j.simpat.2016.07.006.

[230] T. Prike. Open science, replicability, and transparency in modelling. In J. Bijak, editor, *Towards Bayesian Model-Based Demography: Agency, Complexity and Uncertainty in Migration Studies*, Methodos Series, pages 175–183. Springer International Publishing, Cham, 2022. doi:10.1007/978-3-030-83039-7_10.

[231] M. D. Morris. A class of three-level experimental designs for response surface modeling. *Technometrics*, 42(2):111–121, 2000. doi:10.1080/00401706.2000.10485990.

[232] S. Marino, I. B. Hogue, C. J. Ray, and D. E. Kirschner. A methodology for performing global uncertainty and sensitivity analysis in systems biology. *Journal of Theoretical Biology*, 254(1):178–196, 2008. doi:10.1016/j.jtbi.2008.04.011.

[233] P. Mohagheghi and V. Dehlen. Where is the proof? - a review of experiences from applying mde in industry. In *Model Driven Architecture – Foundations and Applications*, pages 432–443, 2008. doi:10.1007/978-3-540-69100-6_31.

[234] R. J. Rodríguez, S. Bernardi, and A. Zimmermann. An evaluation framework for comparative analysis of generalized stochastic petri net simulation techniques. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(8):2834–2844, 2020.

[235] P. Palensky, E. Widl, and A. Elsheikh. Simulating cyber-physical energy systems: Challenges, tools and methods. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(3):318–326, 2014. doi:10.1109/TSMCC.2013.2265739.

[236] V. Grimm, J. Augusiak, A. Focks, B. M. Frank, F. Gabsi, A. S. Johnston, C. Liu, B. T. Martin, M. Meli, V. Radchuk, P. Thorbek, and S. F. Railsback. Towards better modelling and decision support: Documenting model development, testing, and analysis using TRACE. *Ecological Modelling*, 280:129–139, 2014. doi:10.1016/j.ecolmodel.2014.01.018.

[237] P. Wilsdorf, A. Wolpers, J. Hilton, F. Haack, and A. Uhrmacher. Automatic reuse, adaption, and execution of simulation experiments via provenance patterns. *ACM Transactions on Modeling and Computer Simulation*, 33(1–2), 2023. doi:10.1145/3564928.

[238] P. Wilsdorf, N. Fischer, F. Haack, and A. M. Uhrmacher. Exploiting provenance and ontologies in supporting best practices for simulation experiments: A case study on sensitivity analysis. In *2021 Winter Simulation Conference (WSC)*, pages 1–12. IEEE, 2021. doi:10.1109/WSC52266.2021.9715362.

[239] A. Hocquet and F. Wieber. Epistemic issues in computational reproducibility: software as the elephant in the room. *European Journal for Philosophy of Science*, 11:1–20, 2021. doi:10.1007/s13194-021-00362-9.

[240] B. P. Zeigler. *Multifacetted Modelling and Discrete Event Simulation*. Academic Press Professional, Inc., 1984.

[241] J. Hillston. A tool to enhance model exploitation. *Performance Evaluation*, 22(1):59–74, 1995. doi:10.1016/0166-5316(93)E0038-7.

[242] J. Himmelspach, R. Ewald, and A. M. Uhrmacher. A flexible and scalable experimentation layer. In *Proceedings of the 40th Conference on Winter Simulation (WSC)*, pages 827–835. IEEE, 2008. doi:10.1109/WSC.2008.4736146.

[243] K. Pawlikowski, H.-D. Jeong, and J.-S. Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications Magazine*, 40(1):132–139, 2002. doi:10.1109/35.978060.

[244] J. Denil, S. Klikovits, P. J. Mosterman, A. Vallecillo, and H. Vangheluwe. The experiment model and validity frame in M&S. In *Proceedings of the Symposium on Theory of Modeling & Simulation*, pages 1–12, 2017. doi:10.5555/3108905.3108915.

[245] H. Rahmandad and J. D. Sterman. Reporting guidelines for simulation-based research in social sciences. *Systems Dynamics Review*, 28(4):396–411, 2012. doi:10.1002/sdr.1481.

[246] T. Monks, C. S. M. Currie, B. S. Onggo, S. Robinson, M. Kunc, and S. J. E. Taylor. Strengthening the reporting of empirical simulation studies: Introducing the stress guidelines. *Journal of Simulation*, 13(1):55–67, 2019. doi:10.1080/17477778.2018.1442155.

[247] D. Waltemath, R. Adams, D. A. Beard, F. T. Bergmann, U. S. Bhalla, R. Britten, et al. Minimum information about a simulation experiment (MIASE). *PLOS Computational Biology*, 7(4):1–4, 2011. doi:10.1371/journal.pcbi.1001122.

[248] J. Schützel, D. Peng, A. M. Uhrmacher, and L. F. Perrone. Perspectives on languages for specifying simulation experiments. In *2014 Winter Simulation Conference (WSC)*, pages 2836–2847. IEEE, 2014. doi:10.1109/WSC.2014.7020125.

[249] J. Liang, Z. Lin, and Y. Ma. OF-NEDL: an openflow networking experiment description language based on XML. In *Web Information Systems and Mining: International Conference, WISM 2012, Chengdu, China*, pages 686–697. Springer, 2012. doi:10.1007/978-3-642-33469-6_85.

[250] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016. doi:10.1038/sdata.2016.18.

[251] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, et al. Jupyter notebooks - a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, volume 2016. IOS Press, 2016. doi:10.3233/978-1-61499-649-1-87.

[252] P. Wilsdorf, A. W. Kirchhübel, and A. M. Uhrmacher. Nbsimgen: Jupyter notebook extension for generating simulation experiments. In *2023 Winter Simulation Conference (WSC)*, pages 2884–2895, 2023. doi:10.1109/WSC60868.2023.10408537.

[253] F. T. Bergmann, R. Adams, S. Moodie, J. Cooper, M. Glont, M. Golebiewski, M. Hucka, C. Laibe, A. K. Miller, D. P. Nickerson, et al. COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. *BMC Bioinformatics*, 15(1): 1–9, 2014. doi:10.1186/s12859-014-0369-z.

[254] L. A. Carvalho, K. Belhajjame, and C. B. Medeiros. Converting scripts into reproducible workflow research objects. In *2016 IEEE 12th International Conference on e-Science (e-Science)*, pages 71–80. IEEE, 2016. doi:10.1109/eScience.2016.7870887.

[255] D. H. Ton That, G. Fils, Z. Yuan, and T. Malik. Sciunits: Reusable research objects. In *2017 IEEE 13th International Conference on e-Science (e-Science)*, pages 374–383, 2017. doi:10.1109/eScience.2017.51.

[256] M. A. Janssen, L. N. Alessa, M. Barton, S. Bergin, and A. Lee. Towards a community framework for agent-based modelling. *Journal of Artificial Societies and Social Simulation*, 11(2):6, 2008. URL `https://www.jasss.org/11/2/6.html`. Last accessed 19 July, 2024.

[257] D. Cetinkaya and A. Verbraeck. Metamodeling and model transformations in modeling and simulation. In *2011 Winter Simulation Conference (WSC)*, pages 3043–3053, 2011. doi:10.1109/WSC.2011.6148005.

[258] G. Guizzardi and G. Wagner. Conceptual simulation modeling with Onto-UML advanced tutorial. In *2012 Winter Simulation Conference (WSC)*, pages 1–15. IEEE, 2012. doi:10.1109/WSC.2012.6465133.

[259] F. Santos, I. Nunes, and A. L. Bazzan. Quantitatively assessing the benefits of model-driven development in agent-based modeling and simulation. *Simulation Modelling Practice and Theory*, 104:102126, 2020. doi:10.1016/j.simpat.2020.102126.

[260] G.-D. Kapos, A. Tsadimas, C. Kotronis, V. Dalakas, M. Nikolaidou, and D. Anagnostopoulos. A declarative approach for transforming sysml models to executable simulation models. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–16, 2019. doi:10.1109/TSMC.2019.2922153.

[261] L. Capocchi, J. F. Santucci, T. Pawletta, H. Folkerts, and B. P. Zeigler. Discrete-event simulation model generation based on activity metrics. *Simulation Modelling Practice and Theory*, 103:102122, 2020. doi:10.1016/j.simpat.2020.102122.

[262] P. Bocciarelli, A. D'Ambrogio, A. Falcone, A. Garro, and A. Giglio. A model-driven approach to enable the simulation of complex systems on distributed architectures. *Simulation*, 95(12):1185–1211, 2019. doi:10.1177/0037549719829828.

[263] H. Vangheluwe and J. de Lara. Meta-models are models too. In *Winter Simulation Conference*, volume 1, pages 597–605, 2002. doi:10.1109/WSC.2002.1172936.

[264] P. A. Fishwick and J. A. Miller. Ontologies for modeling and simulation: Issues and approaches. In *2004 Winter Simulation Conference (WSC)*, page 264, 2004. doi:10.1109/WSC.2004.1371324.

[265] S. J. Taylor, D. Bell, N. Mustafee, S. de Cesare, M. Lycett, and P. A. Fishwick. Semantic web services for simulation component reuse and interoperability: An ontology approach. In *Organizational advancements through enterprise information systems: Emerging applications and developments*. IGI Global, 2010. doi:10.4018/978-1-60566-968-7.ch021.

[266] G. A. Silver, J. A. Miller, M. Hybinette, G. Baramidze, and W. S. York. Demo: an ontology for discrete-event modeling and simulation. *SIMULATION*, 87(9):747–773, 2011. doi:10.1177/0037549710386843.

[267] H. Cheong and A. Butscher. Physics-based simulation ontology: an ontology to support modelling and reuse of data for physics-based simulation. *Journal of Engineering Design*, 30(10-12):655–687, 2019. doi:10.1080/09544828.2019.1644301.

[268] J. Whittle, J. Hutchinson, and M. Rouncefield. The state of practice in model-driven engineering. *IEEE Software*, 31(3):79–85, 2014. doi:10.1109/MS.2013.65.

[269] J. Bezivin and O. Gerbe. Towards a precise definition of the OMG/MDA framework. In *16th Annual International Conference on Automated Software Engineering (ASE 2001)*, pages 273–280, 2001. doi:10.1109/ASE.2001.989813.

[270] ISO. ISO/IEC 14977:1996(E) information technology - syntactic metalanguage - extended BNF. 1996.

[271] Unified modeling language, 2020. Available online: `https://www.omg.org/spec/UML`, last accessed 19 July, 2024.

[272] D. C. Fallside and P. Walmsley. XML schema part 0: primer second edition. *W3C Recommendation*, 2004. URL `https://www.w3.org/TR/xmlschema-0`. Last accessed 19 July, 2024.

[273] JSON Schema. JSON Schema Draft-07 Release Notes, 2018. URL `https://json-schema.org/draft-07/json-schema-release-notes.html`. Last accessed 19 July, 2024.

[274] M. Hucka, F. T. Bergmann, C. Chaouiya, A. Dräger, S. Hoops, S. M. Keating, et al. The systems biology markup language (SBML): Language specification for level 3 version 2 core release 2. *Journal of Integrative Bioinformatics*, 16(2), 2019. doi:10.1515/jib-2019-0021.

[275] D. D. F. Marcos, B. Jean, J. Frédéric, B. Erwan, and G. Guillaume. AMW: A generic model weaver. *Proc. of the 1eres Journées sur l'Ingénierie Dirigée par les Modeles*, 200, 2005. URL `http://www.sciences.univ-nantes.fr/lina/atl/www/papers/IDM_2005_weaver.pdf`. Last accessed 19 July, 2024.

[276] L. Bettini, D. Di Ruscio, L. Iovino, and A. Pierantonio. Supporting safe metamodel evolution with edelta. *International Journal on Software Tools for Technology Transfer*, 24 (2):247–260, 2022. doi:10.1007/s10009-022-00646-2.

[277] M. Haeusler, T. Trojer, J. Kessler, M. Farwick, E. Nowakowski, and R. Breu. Combining versioning and metamodel evolution in the chronosphere model repository. In A. Tjoa, L. Bellatreche, S. Biffl, J. van Leeuwen, and J. Wiedermann, editors, *International Conference on Current Trends in Theory and Practice of Informatics*, Lecture Notes in Computer Science, pages 153–167. Springer Cham, 2018. doi:10.1007/978-3-319-73117-9_11.

[278] B. Liu, V. P. Betancourt, T. Glock, M. Kern, E. Sax, and J. Becker. Model-driven design of tools for multi-domain systems with loosely coupled metamodels. In *2019 IEEE International Systems Conference (SysCon)*, pages 1–7, 2019. doi:10.1109/SYSCON.2019.8836952.

[279] P. Dhar, T. C. Meng, S. Somani, L. Ye, A. Sairam, M. Chitre, Z. Hao, and K. Sakharkar. Cellware—a multi-algorithmic software for computational systems biology. *Bioinformatics*, 20(8):1319–1321, 2004. doi:10.1093/bioinformatics/bth067.

[280] M. Bork, L. Geiger, C. Schneider, and A. Zündorf. Towards roundtrip engineering - a template-based reverse engineering approach. In I. Schieferdecker and A. Hartman, editors, *Model Driven Architecture – Foundations and Applications*, Lecture Notes in Computer Science, pages 33–47. Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-69100-6_3.

[281] J. C. Thiele. R marries NetLogo: Introduction to the RNetLogo package. *Journal of Statistical Software*, 58(2):1–41, 2014. doi:10.18637/jss.v058.i02.

[282] S. Tisue and U. Wilensky. NetLogo: A simple environment for modeling complexity. In *International Conference on Complex Systems*, volume 21, pages 16–21. Citeseer, 2004. URL `http://ccl.northwestern.edu/papers/netlogo-iccs2004.pdf`. Last accessed 19 July, 2024.

[283] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003. doi:10.1093/bioinformatics/btg015.

[284] T. Warnke and A. M. Uhrmacher. Complex simulation experiments made easy. In *2018 Winter Simulation Conference (WSC)*, pages 410–424. IEEE, 2018. doi:10.1109/WSC.2018.8632429.

[285] T. Székely Jr and K. Burrage. Stochastic simulation in systems biology. *CComputational and Structural Biotechnology Journal*, 12(20-21):14–25, 2014. doi:10.1016/j.csbj.2014.10.003.

[286] T. Warnke, O. Reinhardt, A. Klabunde, F. Willekens, and A. M. Uhrmacher. Modelling and simulating decision processes of linked lives: An approach based on concurrent processes and stochastic race. *Population Studies*, 71(sup1):69–83, 2017. doi:10.1080/00324728.2017.1380960.

[287] S. Cauchemez and N. M. Ferguson. Likelihood-based estimation of continuous-time epidemic models from time-series data: application to measles transmission in London. *Journal of The Royal Society Interface*, 5(25):885–897, 2008. doi:10.1098/rsif.2007.1292.

[288] R. E. Caflisch et al. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, 1998: 1–49, 1998. doi:10.1017/S0962492900002804.

[289] T. Crestaux, O. Le Maıtre, and J.-M. Martinez. Polynomial chaos expansion for sensitivity analysis. *Reliability Engineering and System Safety*, 94(7):1161–1172, 2009. doi:10.1016/j.ress.2008.10.008.

[290] I. M. Sobol. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55(1):271–280, 2001. doi:10.1016/S0378-4754(00)00270-6.

[291] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola. Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. *Computer Physics Communications*, 181(2):259–270, 2010. doi:10.1016/j.cpc.2009.09.018.

[292] M. Vasilevski, E. Queiroz, A. L. Fonseca, I. Silva, S. Y. Catunda, and L. A. Guedes. Systemc ams modeling of a sensor node energy consumption and battery state-of-charge for wsn. In *2015 IEEE 13th International New Circuits and Systems Conference (NEWCAS)*, pages 1–4, 2015. doi:10.1109/NEWCAS.2015.7181996.

[293] I. M. Sobol, D. Asotsky, A. Kreinin, and S. Kucherenko. Construction and comparison of high-dimensional sobol' generators. *Wilmott*, 2011(56):64–79, 2011. doi:10.1002/wilm.10056.

[294] S. Kucherenko, D. Albrecht, and A. Saltelli. Exploring multi-dimensional spaces: A comparison of latin hypercube and quasi monte carlo sampling techniques. *arXiv preprint arXiv:1505.02350*, 2015. doi:10.48550/arXiv.1505.02350.

[295] A. Singhee and R. A. Rutenbar. Why quasi-monte carlo is better than monte carlo or latin hypercube sampling for statistical circuit analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(11):1763–1776, 2010. doi:10.1109/TCAD.2010.2062750.

[296] T. L. Yibo Sun, Xiuyun Meng and Y. Wu. A fast optimal latin hypercube design method using an improved translational propagation algorithm. *Engineering Optimization*, 52(7): 1244–1260, 2020. doi:10.1080/0305215X.2019.1642881.

[297] T. U. Consortium. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515, 2018. doi:10.1093/nar/gky1049.

[298] J. Schöberl. NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science*, 1(1):41–52, 1997. doi:10.1007/s007910050004.

[299] A. Ruscheinski, P. Wilsdorf, M. Dombrowsky, and A. M. Uhrmacher. Capturing and reporting provenance information of simulation studies based on an artifact-based workflow approach. In *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '19, pages 185–196. ACM, 2019. doi:10.1145/3316480.3325514.

[300] A. Ruscheinski, A. Wolpers, P. Henning, P. Wilsdorf, and A. M. Uhrmacher. SIMPROV: Provenance capturing for simulation studies. In Preparation.

[301] P. Wilsdorf, O. Reinhardt, T. Prike, M. Hinsch, J. Bijak, and A. M. Uhrmacher. Simulation studies of social systems – telling the story based on provenance patterns. *Royal Society Open Science*, 11(8):240258, 2024. doi:10.1098/rsos.240258.

[302] P. Wilsdorf and A. M. Uhrmacher. Creating PROV-DM graphs from model databases. In *2022 Winter Simulation Conference (WSC)*, pages 2118–2129. IEEE, 2022. doi:10.1109/WSC57314.2022.10015331.

[303] K.-D. Schewe and B. Thalheim. Conceptual modelling of web information systems. *Data & Knowledge Engineering*, 54(2):147–188, 2005. doi:10.1016/j.datak.2004.08.005.

[304] I. Davies, P. Green, M. Rosemann, M. Indulska, and S. Gallo. How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, 58(3):358–380, 2006. doi:10.1016/j.datak.2005.07.007.

[305] B. Thalheim. Towards a theory of conceptual modelling. In C. A. Heuser and G. Pernul, editors, *Advances in Conceptual Modeling - Challenging Perspectives*, Lecture Notes in Computer Science, pages 45–54. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-04947-7_7.

[306] R. E. Nance. The conical methodology and the evolution of simulation model development. *Annals of Operations Research*, 53(1):1–45, 1994. doi:10.1007/BF02136825.

[307] S. Robinson. Conceptual modelling for simulation part II: a framework for conceptual modelling. *Journal of the Operational Research Society*, 59(3):291–304, 2008. doi:10.1057/palgrave.jors.2602369.

[308] D. W. Embley and B. Thalheim. *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges*. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-15865-0.

[309] O. Balci, J. D. Arthur, and W. F. Ormsby. Achieving reusability and composability with a simulation conceptual model. *Journal of Simulation*, 5(3):157–165, 2011. doi:10.1057/jos.2011.7.

[310] S. Robinson, G. Arbez, L. G. Birta, A. Tolk, and G. Wagner. Conceptual modeling: definition, purpose and benefits. In *2015 Winter Simulation Conference (WSC)*, pages 2812–2826. IEEE, 2015. doi:10.1109/WSC.2015.7408386.

[311] N. A. Jones, H. Ross, T. Lynam, P. Perez, and A. Leitch. Mental models: an interdisciplinary synthesis of theory and methods. *Ecology and society*, 16(1), 2011. URL `https://www.jstor.org/stable/26268859`. Last accessed 19 July, 2024.

[312] P. A. Fishwick. Simulation model design. In *Proceedings of the 27th Conference on Winter Simulation*, WSC '95, page 209–211. IEEE Computer Society, 1995. doi:10.1145/224401.224464.

[313] D. Cetinkaya, A. Verbraeck, and M. D. Seck. Model transformation from BPMN to DEVS in the MDD4MS framework. In *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium*, San Diego, CA, 2012. Society for Computer Simulation International. ISBN 9781618397867.

[314] D. A. Hollmann, M. Cristiá, and C. Frydman. CML-DEVS: A specification language for DEVS conceptual models. *Simulation Modelling Practice and Theory*, 57:100–117, 2015. doi:10.1016/j.simpat.2015.06.007.

[315] A. Guru and P. Savory. A template-based conceptual modeling infrastructure for simulation of physical security systems. In *Proceedings of the 2004 Winter Simulation Conference*, volume 1, page 873. IEEE, 2004. doi:10.1109/WSC.2004.1371401.

[316] N. Furian, M. O'Sullivan, C. Walker, S. Vössner, and D. Neubacher. A conceptual modeling framework for discrete event simulation using hierarchical control structures. *Simulation Modelling Practice and Theory*, 56:82–96, 2015. doi:10.1016/j.simpat.2015.04.004.

[317] G. Arbez and L. G. Birta. The ABCmod conceptual modeling framework. In *Conceptual Modeling for Discrete-event Simulation*, pages 149–194. CRC Press, 2010. ISBN 9780429148866.

[318] D. K. Pace. Ideas about simulation conceptual model development. *Johns Hopkins APL Technical Digest*, 21(3):327–336, 2000. URL `https://www.jhuapl.edu/Content/techdigest/pdf/V21-N03/21-03-Pace2.pdf`. Last accessed 19 July, 2024.

[319] L. Chwif, J. Banks, J. P. de Moura Filho, and B. Santini. A framework for specifying a discrete-event simulation conceptual model. *Journal of Simulation*, 7(1):50–60, 2013. doi:10.1057/jos.2012.18.

[320] N. L. Novère, A. Finney, M. Hucka, U. S. Bhalla, F. Campagne, J. Collado-Vides, E. J. Crampin, M. Halstead, E. Klipp, P. Mendes, et al. Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature Biotechnology*, 23(12):1509–1515, 2005. doi:10.1038/nbt1156.

[321] The Gene Ontology Consortium. The gene ontology resource: 20 years and still GOing strong. *Nucleic Acids Research*, 47(D1):D330–D338, 2018. doi:10.1093/nar/gky1055.

[322] S. Sarntivijai, Y. Lin, Z. Xiang, T. F. Meehan, A. D. Diehl, U. D. Vempati, S. C. Schürer, C. Pang, J. Malone, H. Parkinson, et al. CLO: the cell line ontology. *Journal of biomedical semantics*, 5(1):1–10, 2014. doi:10.1186/2041-1480-5-37.

[323] MAMO. 2022. Mathematical Modelling Ontology. `https://bioportal.bioontology.org/ontologies/MAMO`, last accessed 19 July, 2024.

[324] S. Achter, M. Borit, E. Chattoe-Brown, and P.-O. Siebers. RAT-RS: A reporting standard for improving the documentation of data use in agent-based modelling. *International Journal of Social Research Methodology*, 25(4):517–540, 2022. doi:10.1080/13645579.2022.2049511.

[325] N. P. C. Hong, D. S. Katz, M. Barker, A.-L. Lamprecht, C. Martinez, F. E. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, et al. FAIR principles for research software (FAIR4RS principles). *Research Data Alliance*, 2022. doi:10.15497/RDA00065.

[326] D. Ayllón, S. F. Railsback, C. Gallagher, J. Augusiak, H. Baveco, U. Berger, S. Charles, R. Martin, A. Focks, N. Galic, C. Liu, E. E. van Loon, et al. Keeping modelling notebooks with TRACE: Good for you and good for environmental research and management support. *Environmental Modelling & Software*, 136:104932, 2021. doi:10.1016/j.envsoft.2020.104932.

[327] A. Van Deursen and P. Klint. Domain-specific language design requires feature descriptions. *Journal of Computing and Information Technology*, 10(1):1–17, 2002. doi:10.2498/cit.2002.01.01.

[328] M. Fowler. *Domain-specific Languages*. Addison-Wesley Professional, 2010. ISBN 9780132107549.

[329] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13(1):45–60, 1981. doi:10.1016/0304-3975(81)90110-9.

[330] F. Lorig, C. A. Becker, and I. J. Timm. Formal specification of hypotheses for assisting computer simulation studies. In *Proceedings of the Symposium on Theory of Modeling & Simulation*, pages 1–12, San Diego, CA, 2017. Society for Computer Simulation International. doi:10.5555/3108905.3108923.

[331] S. Wolny, A. Mazak, C. Carpella, V. Geist, and M. Wimmer. Thirteen years of SysML: a systematic mapping study. *Software and Systems Modeling*, 19(1):111–169, 2020. doi:10.1007/s10270-019-00735-y.

[332] H. Elmqvist, S. E. Mattsson, and M. Otter. Object-oriented and hybrid modeling in modelica. *Journal Européen des systèmes automatisés*, 35(4):395–404, 2001. URL `https://elib.dlr.de/11522/`. Last accessed 19 July, 2024.

[333] S. A. White. Introduction to BPMN. *IBM Cooperation*, 2(0):0, 2004.

[334] N. L. Novère, M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M. I. Aladjem, S. M. Wimalaratne, F. T. Bergman, R. Gauges, et al. The Systems Biology Graphical Notation. *Nature Biotechnology*, 27(8):735–741, 2009. doi:10.1038/nbt.1558.

[335] M. Courtot, N. Juty, C. Knüpfer, D. Waltemath, A. Zhukova, A. Dräger, M. Dumontier, A. Finney, M. Golebiewski, J. Hastings, et al. Controlled vocabularies and semantics in systems biology. *Molecular Systems Biology*, 7(1):543, 2011. doi:10.1038/msb.2011.77.

[336] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1):161–197, 1998. doi:10.1016/S0169-023X(97)00056-6.

[337] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, Lecture Notes in Computer Science, pages 722–735. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-76298-0_52.

[338] V. Petri, P. Jayaraman, M. Tutaj, G. T. Hayman, J. R. Smith, J. De Pons, S. J. Laulederkind, T. F. Lowry, R. Nigam, S.-J. Wang, M. Shimoyama, M. R. Dwinell, et al. The pathway ontology – updates and applications. *Journal of Biomedical Semantics*, 5(1):7, 2014. doi:10.1186/2041-1480-5-7.

[339] M. Gremse, A. Chang, I. Schomburg, A. Grote, M. Scheer, C. Ebeling, and D. Schomburg. The BRENDA tissue ontology (BTO): the first all-integrating ontology of all organisms for enzyme sources. *Nucleic Acids Research*, 39(suppl_1):D507–D513, 2011. doi:10.1093/nar/gkq968.

[340] F. Boussuge, C. M. Tierney, H. Vilmart, T. T. Robinson, C. G. Armstrong, D. C. Nolan, J.-C. Léon, and F. Ulliana. Capturing simulation intent in an ontology: CAD and CAE integration application. *Journal of Engineering Design*, 30(10-12):688–725, 2019. doi:10.1080/09544828.2019.1630806.

[341] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins. The ontology of tasks and methods. In *AAAI Technical Report SS-97-06*. 1998.

[342] J. Malone, E. Holloway, T. Adamusiak, M. Kapushesky, J. Zheng, N. Kolesnikov, A. Zhukova, A. Brazma, and H. Parkinson. Modeling sample variables with an Experimental Factor Ontology. *Bioinformatics*, 26(8):1112–1118, 2010. doi:10.1093/bioinformatics/btq099.

[343] J. M. Keil and S. Schindler. Comparison and evaluation of ontologies for units of measurement. *Semantic Web*, 10(1):33–51, Jan. 2019. ISSN 1570-0844. doi:10.3233/SW-180310. URL `https://content.iospress.com/articles/semantic-web/sw310`. Publisher: IOS Press.

[344] N. F. Noy, D. L. McGuinness, and others. Ontology development 101: A guide to creating your first ontology. 2001. URL `http://www.ksl.stanford.edu/KSL_Abstracts/KSL-01-05.html`. Stanford knowledge systems laboratory technical report KSL-01-05, last accessed 19 July, 2024.

[345] F. N. Al-Aswadi, H. Y. Chan, and K. H. Gan. Automatic ontology construction from text: a review from shallow to deep learning trend. *Artificial Intelligence Review*, 53(6): 3901–3928, 2020. doi:10.1007/s10462-019-09782-9.

[346] A. T. Bittig and A. M. Uhrmacher. Spatial modeling in cell biology at multiple levels. In *2010 Winter Simulation Conference (WSC)*, pages 608–619. IEEE, 2010. doi:10.1109/WSC.2010.5679125.

[347] G. J. Klir and I. Rozehnal. Epistemological categories of systems: An overview and mathematical formulation. In F. Pichler and R. Moreno-Diaz, editors, *Computer Aided Systems Theory — EUROCAST '89*, pages 7–32. Springer Berlin Heidelberg, 1990. ISBN 978-3-540-46932-2.

[348] A. G. Hoekstra, B. Chopard, D. Coster, S. Portegies Zwart, and P. V. Coveney. Multiscale computing for science and engineering in the era of exascale performance. *Philosophical Transactions of the Royal Society A*, 377(2142):20180144, 2019. doi:10.1098/rsta.2018.0144.

[349] E. H. Page. *Simulation Modeling Methodology: Principles and Etiology of Decision Support*. PhD thesis, Virginia Polytechnic Institute and State University, 1994. URL `https://vtechworks.lib.vt.edu/items/12b1c253-2cc6-4efc-b60c-6a1a80d3e970`. Last accessed 19 July, 2024.

[350] B. P. Zeigler, H. S. Song, T. G. Kim, and H. Praehofer. Devs framework for modelling, simulation, analysis, and design of hybrid systems. In *Hybrid Systems II*, pages 529–551. Springer Berlin Heidelberg, 1995. ISBN 978-3-540-47519-4.

[351] O. Dalle. On reproducibility and traceability of simulations. In *2012 Winter Simulation Conference (WSC)*, pages 1–12. IEEE, 2012. doi:10.1109/WSC.2012.6465284.

[352] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004. doi:10.1093/bioinformatics/bth361.

[353] H. Sakane, H. Yamamoto, and A. Kikuchi. LRP6 is internalized by Dkk1 to suppress its phosphorylation in the lipid raft and is recycled for reuse. *Journal of Cell Science*, 123(3): 360–368, 2010. doi:10.1242/jcs.058008.

[354] M. Krötzsch, D. Vrandečić, and M. Völkel. Semantic MediaWiki. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. M. Aroyo, editors, *The Semantic Web - ISWC 2006*, pages 935–942. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-49055-5.

[355] T. Meyer, A. Ruscheinski, P. Wilsdorf, and A. M. Uhrmacher. Simulation-supported engineering of self-adaptive software systems. In *2021 Winter Simulation Conference (WSC)*, pages 1–12, 2021. doi:10.1109/WSC52266.2021.9715324.

[356] S. Rai, R. C. Belwal, and A. Gupta. A review on source code documentation. *ACM Transactions on Intelligent Systems and Technology*, 13(5), 2022. doi:10.1145/3519312.

[357] J. Y. Khan and G. Uddin. Automatic code documentation generation using GPT-3. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22. ACM, 2023. doi:10.1145/3551349.3559548.

[358] I. Jackson and M. J. Saenz. From natural language to simulations: Applying GPT-3 codex to automate simulation modeling of logistics systems. *arXiv preprint arXiv:2202.12107*, 2022.

[359] L. Moreau and P. Groth. Provenance: An introduction to PROV. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 3(4):1–129, 2013. doi:10.2200/S00528ED1V01Y201308WBE007.

[360] O. Reinhardt, A. Rucheinski, and A. M. Uhrmacher. ODD+P: Complementing the ODD protocol with provenance information. In *2018 Winter Simulation Conference (WSC)*, pages 727–738. IEEE, 2018. doi:10.1109/WSC.2018.8632481.

[361] I. Altintas, O. Barney, and E. Jaeger-Frank. Provenance collection support in the Kepler scientific workflow system. In *International Provenance and Annotation Workshop*, pages 118–132. Springer Berlin Heidelberg, 2006. doi:10.1007/11890850_14.

[362] L. Murta, V. Braganholo, F. Chirigati, D. Koop, and J. Freire. noWorkflow: Capturing and analyzing provenance of scripts. In *Provenance and Annotation of Data and Processes*, pages 71–83. Springer Cham, 2015. ISBN 978-3-319-16462-5.

[363] T. M. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, K. Bocinsky, Y. Cao, F. Chirigati, S. C. Dey, J. Freire, D. N. Huntzinger, C. Jones, et al. YesWorkflow: A user-oriented, language-independent tool for recovering workflow information from scripts. *arXiv*, 2015. doi:10.48550/arXiv.1502.02403. Preprint.

[364] M. Schröder, S. Staehlke, P. Groth, J. B. Nebe, S. Spors, and F. Krüger. Structure-based knowledge acquisition from electronic lab notebooks for research data provenance documentation. *Journal of Biomedical Semantics*, 13(1):1–22, 2022. doi:10.1186/s13326-021-00257-x.

[365] C. Triebig and F. Klügl. Elements of a documentation framework for agent-based simulation models. *Cybernetics and Systems*, 40(5):441–474, 2009. doi:10.1080/01969720902922459.

[366] E. v. Elm, D. G. Altman, M. Egger, S. J. Pocock, P. C. Gøtzsche, and J. P. Vandenbroucke. The strengthening the reporting of observational studies in epidemiology (STROBE) statement: guidelines for reporting observational studies. *The Lancet*, 370(9596):1453–1457, 2007. doi:10.1016/S0140-6736(07)61602-X.

[367] K. Görlach, M. Sonntag, D. Karastoyanova, F. Leymann, and M. Reiter. Conventional Workflow Technology for Scientific Simulation. In X. Yang, L. Wang, and W. Jie, editors, *Guide to e-Science: Next Generation Scientific Research and Discovery*, pages 323–352. Springer London, 2011. doi:10.1007/978-0-85729-439-5_12.

[368] R. Barga and D. Gannon. Scientific versus Business Workflows. In I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, editors, *Workflows for e-Science: Scientific Workflows for Grids*, pages 9–16. Springer, London, 2007. doi:10.1007/978-1-84628-757-2_2.

[369] M. Chinosi and A. Trombetta. BPMN: An introduction to the standard. *Computer Standards & Interfaces*, 34(1):124–134, 2012. doi:10.1016/j.csi.2011.06.002.

[370] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pages 423–424, 2004. doi:10.1109/SSDM.2004.1311241.

[371] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, and others. The Taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic acids research*, 41(W1):W557–W561, 2013. doi:10.1093/nar/gkt328.

[372] G. De Giacomo, M. Dumas, F. M. Maggi, and M. Montali. Declarative Process Modeling in BPMN. In J. Zdravkovic, M. Kirikova, and P. Johannesson, editors, *Advanced Information Systems Engineering*, Lecture Notes in Computer Science, pages 84–100. Springer Cham, 2015. doi:10.1007/978-3-319-19069-3_6.

[373] R. Hull, E. Damaggio, R. De Masellis, F. Fournier, M. Gupta, F. T. Heath, III, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. N. Sukaviriya, and R. Vaculin. Business artifacts with guard-stage-milestone lifecycles: Managing artifact interactions with conditions and events. In *Proceedings of the 5th ACM International Conference on Distributed Event-based System*, DEBS '11, pages 51–62. ACM, 2011. doi:10.1145/2002259.2002270.

[374] W. M. van Der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development*, 23:99–113, 2009. doi:10.1007/s00450-009-0057-9.

[375] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi. Prospective and retrospective provenance collection in scientific workflow environments. In *2010 IEEE International Conference on Services Computing*, pages 449–456, 2010. doi:10.1109/SCC.2010.18.

[376] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson. The Open Provenance Model: An overview. In *Provenance and Annotation of Data and Processes*, pages 323–326. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-89965-5.

[377] K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, et al. PROV-DM: The PROV data model. *W3C Recommendation*, 2013. URL `https://www.w3.org/TR/prov-dm/`. Last accessed 19 July, 2024.

[378] I. Taxidou, T. De Nies, and P. M. Fischer. Provenance of explicit and implicit interactions on social media with W3C PROV-DM. In M. Atzmueller, A. Chin, F. Lemmerich, and C. Trattner, editors, *Behavioral Analytics in Social and Ubiquitous Environments*, Lecture Notes in Computer Science, pages 126–150. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-34407-8_7.

[379] L. Sadineni, E. S. Pilli, and R. B. Battula. Ready-IoT: A novel forensic readiness model for internet of things. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, pages 89–94, 2021. doi:10.1109/WF-IoT51360.2021.9595902.

[380] A. Ruscheinski, D. Gjorgevikj, M. Dombrowsky, K. Budde, and A. M. Uhrmacher. Towards a PROV ontology for simulation models. In K. Belhajjame, A. Gehani, and P. Alper, editors, *Provenance and Annotation of Data and Processes*, Lecture Notes in Computer Science, pages 192–195. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-98379-0_17.

[381] M. E. Pierce. *Integrating knowledge about complex adaptive systems: insights from modelling the Eastern Baltic cod*. Universität Rostock, 2022. doi:10.18453/rosdok_id00004121. PhD thesis.

[382] J. J. Miller. Graph database applications and concepts with Neo4j. In *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, volume 2324, 2013. URL `https://aisel.aisnet.org/sais2013/24`. Last accessed 19 July, 2024.

[383] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor. Cypher: an evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1433–1445, 2018. doi:10.1145/3183713.3190657.

[384] M. Herschel, R. Diestelkämper, and H. B. Lahmar. A survey on provenance: What for? what form? what from? *The VLDB Journal*, 26:881–906, 2017. doi:10.1007/s00778-017-0486-1.

[385] M. K. Anand, S. Bowers, and B. Ludäscher. Provenance browser: Displaying and querying scientific workflow provenance graphs. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 1201–1204, 2010. doi:10.1109/ICDE.2010.5447741.

[386] M. D. Allen, L. Seligman, B. Blaustein, and A. Chapman. Provenance capture and use: A practical guide. *The MITRE Corporation*, 2010. URL `https://www.mitre.org/sites/default/files/publications/practical-provenance-guide-MP100128.pdf`. Last accessed 19 July, 2024.

[387] S. M. S. d. Cruz, M. L. M. Campos, and M. Mattoso. Towards a taxonomy of provenance in scientific workflow management systems. In *2009 Congress on Services - I*, pages 259–266, 2009. doi:10.1109/SERVICES-I.2009.18.

[388] K. Belhajjame, K. Wolstencroft, O. Corcho, T. Oinn, F. Tanoh, A. William, and C. Goble. Metadata management in the taverna workflow system. In *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 651–656, 2008. doi:10.1109/CCGRID.2008.17.

[389] E. Angelino, D. Yamins, and M. Seltzer. Starflow: A script-centric data analysis environment. In *Provenance and Annotation of Data and Processes: Third International Provenance and Annotation Workshop, IPAW 2010*, pages 236–250. Springer, 2010. doi:10.1007/978-3-642-17819-1_27.

[390] M. Barker, N. P. Chue Hong, D. S. Katz, A.-L. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, et al. Introducing the fair principles for research software. *Scientific Data*, 9(1):622, 2022. doi:10.1038/s41597-022-01710-x.

[391] D. C. Berrios, A. Beheshti, and S. V. Costes. FAIRness and usability for open-access omics data systems. *AMIA Annual Symposium Proceedings*, 2018:232–241, 2018. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6371294/`. Last accessed 19 July, 2024.

[392] C. J. Markiewicz, K. J. Gorgolewski, F. Feingold, R. Blair, Y. O. Halchenko, E. Miller, N. Hardcastle, J. Wexler, O. Esteban, M. Goncavles, A. Jwa, and R. Poldrack. The OpenNeuro resource for sharing of neuroscience data. *eLife*, 10:e71774, 2021. doi:10.7554/eLife.71774.

[393] C. A. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, and D. De Roure. myExperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Research*, 38 (suppl_2):W677–W682, 2010. doi:10.1093/nar/gkq429.

[394] J. K. Medley, K. Choi, M. König, L. Smith, S. Gu, J. Hellerstein, S. C. Sealfon, and H. M. Sauro. Tellurium notebooks—An environment for reproducible dynamical modeling in systems biology. *PLOS Computational Biology*, 14(6):1–24, 2018. doi:10.1371/journal.pcbi.1006220.

[395] C. M. Lloyd, M. D. Halstead, and P. F. Nielsen. CellML: Its future, present and past. *Progress in Biophysics and Molecular Biology*, 85(2):433–450, 2004. doi:10.1016/j.pbiomolbio.2004.01.004.

[396] S. Federhen. The ncbi taxonomy database. *Nucleic Acids Research*, 40(D1):D136–D143, 2011. doi:10.1093/nar/gkr1178.

[397] G. O. Consortium. The gene ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32(suppl_1):D258–D261, 2004. doi:10.1093/nar/gkh036.

[398] M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, and Y. Yamanishi. KEGG for linking genomes to life and the environment. *Nucleic Acids Research*, 36(suppl_1):D480–D484, 2007. doi:10.1093/nar/gkm882.

[399] R. R. Padala, R. Karnawat, S. B. Viswanathan, A. V. Thakkar, and A. B. Das. Cancerous perturbations within the ERK, PI3K/Akt, and Wnt/β-catenin signaling network constitutively activate inter-pathway positive feedback loops. *Molecular BioSystems*, 13:830–840, 2017. doi:10.1039/C6MB00786D.

[400] A. Shahid, M. T. Afzal, M. Abdar, M. E. Basiri, X. Zhou, N. Y. Yen, and J.-W. Chang. Insights into relevant knowledge extraction techniques: a comprehensive review. *The Journal of Supercomputing*, 76:1695–1733, 2020.

[401] L. M. Schriml, J. B. Munro, M. Schor, D. Olley, C. McCracken, V. Felix, J. Baron, R. Jackson, S. Bello, C. Bearer, R. Lichenstein, K. Bisordi, N. C. Dialo, M. Giglio, and C. Greene. The Human Disease Ontology 2022 update. *Nucleic Acids Research*, 50(D1): D1255–D1261, 2021. doi:10.1093/nar/gkab1063.

[402] E. Demir, M. P. Cary, S. Paley, K. Fukuda, C. Lemer, I. Vastrik, G. Wu, P. D'eustachio, C. Schaefer, J. Luciano, et al. The biopax community standard for pathway data sharing. *Nature Biotechnology*, 28(9):935–942, 2010.

[403] S. L. Campbell, J.-P. Chancelier, and R. Nikoukhah. Modeling and simulation in SCILAB. In *Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4*, pages 73–106. Springer, New York, NY, 2010. doi:10.1007/978-1-4419-5527-2_3.

[404] N. Rodriguez, J.-B. Pettit, P. Dalle Pezze, L. Li, A. Henry, M. P. van Iersel, G. Jalowicki, M. Kutmon, K. N. Natarajan, D. Tolnay, et al. The systems biology format converter. *BMC Bioinformatics*, 17(1):1–7, 2016. doi:10.1186/s12859-016-1000-2.

[405] H. S. Packer, A. Chapman, and L. Carr. GitHub2PROV: Provenance for supporting software project management. In T. Moyer and S. Roy, editors, *Proceedings of the 11th International Workshop on Theory and Practice of Provenance*, Philadelphia, PA, 2019. USENIX Association. URL `https://www.usenix.org/conference/tapp2019/presentation/packer`. Last accessed 19 July, 2024.

[406] M. Scharm, O. Wolkenhauer, and D. Waltemath. An algorithm to detect and communicate the differences in computational models describing biological systems. *Bioinformatics*, 32 (4):563–570, 2015. doi:10.1093/bioinformatics/btv484.

[407] M. Scharm, T. Gebhardt, V. Touré, A. Bagnacani, A. Salehzadeh-Yazdi, O. Wolkenhauer, and D. Waltemath. Evolution of computational models in biomodels database and the physiome model repository. *BMC Systems Biology*, 12(1):1–10, 2018. doi:10.1186/s12918-018-0553-2.

[408] M. Feng and J. Staum. Green simulation: reusing the output of repeated experiments. *ACM Transactions on Modeling and Computer Simulation*, 27(4), 2017. doi:10.1145/3129130.

[409] J. Ma, S. Lee, K. W. Cho, and Y.-K. Suh. A simulation provenance data management system for efficient job execution on an online computational science engineering platform. *Cluster Computing*, 22(1):147–159, 2019. doi:10.1007/s10586-018-2827-2.

[410] Y.-K. Suh and K. Y. Lee. A survey of simulation provenance systems: Modeling, capturing, querying, visualization, and advanced utilization. *Human-centric Computing and Information Sciences*, 8(1):27, 2018. doi:10.1186/s13673-018-0150-9.

[411] J. Hillston, A. L. Opdahl, and R. Pooley. A case study using the IMSE experimentation tool. In R. Andersen, J. A. Bubenko, and A. Sølvberg, editors, *Advanced Information Systems Engineering*, pages 284–306, Berlin, Heidelberg, 1991. Springer. ISBN 978-3-540-47378-7. doi:10.1007/3-540-54059-8_90.

[412] L. G. Birta and F. N. Özmizrak. A knowledge-based approach for the validation of simulation models: the foundation. *ACM Transactions on Modeling and Computer Simulation*, 6(1): 76–98, Jan. 1996. ISSN 1049-3301. doi:10.1145/229493.229511. URL `https://doi.org/10.1145/229493.229511`.

[413] P. De Bièvre. The 2012 international vocabulary of metrology: "VIM". *Accreditation and Quality Assurance*, 17(2):231–232, 2012. doi:10.1007/s00769-012-0885-3.

[414] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*, 22(2):67–120, 2012. doi:10.1002/stvr.430.

[415] C. Omar, J. Aldrich, and R. C. Gerkin. Collaborative infrastructure for test-driven scientific model validation. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, page 524–527, 2014. doi:10.1145/2591062.2591129.

[416] N. Popper, M. Zechmeister, D. Brunmeir, C. Rippinger, N. Weibrecht, C. Urach, M. Bicher, G. Schneckenreither, and A. Rauber. Synthetic reproduction and augmentation of covid-19 case reporting data by agent-based simulation. *medRxiv*, 2020. doi:10.1101/2020.11.07.20227462.

[417] R. Randhawa, C. Shaffer, and J. Tyson. Model composition for macromolecular regulatory networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(2): 278–287, 2010. doi:10.1109/TCBB.2008.64.

[418] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe. Co-simulation: A survey. *ACM Computing Surveys*, 51(3), 2018. doi:10.1145/3179993.

[419] P. V. Coveney, D. Groen, and A. G. Hoekstra. Reliability and reproducibility in computational science: implementing validation, verification and uncertainty quantification in silico. *Philosophical Transactions of the Royal Society A*, 379, 2021. doi:10.1098/rsta.2020.0409.

[420] R. Axtell, R. Axelrod, J. M. Epstein, and M. D. Cohen. Aligning simulation models: A case study and results. *Computational & Mathematical Organization Theory*, 1(2):123–141, 1996. doi:10.1007/BF01299065.

[421] D. Hales, J. Rouchier, and B. Edmonds. Model-to-model analysis. *Journal of Artificial Societies and Social Simulation*, 6(4), 2003. URL `https://www.jasss.org/6/4/5.html`. Last accessed 19 July, 2024.

[422] P. Missier, B. Ludäscher, S. Bowers, S. Dey, A. Sarkar, B. Shrestha, I. Altintas, M. K. Anand, and C. Goble. Linking multiple workflow provenance traces for interoperable collaborative science. In *The 5th Workshop on Workflows in Support of Large-Scale Science*, pages 1–8, 2010. doi:10.1109/WORKS.2010.5671861.

[423] T. Gebhardt, V. Touré, D. Waltemath, O. Wolkenhauer, and M. Scharm. Exploring the evolution of biochemical models at the network level. *PLOS ONE*, 17(3):1–15, 2022. doi:10.1371/journal.pone.0265735.

[424] F. Schreiber, B. Sommer, T. Czauderna, M. Golebiewski, T. E. Gorochowski, M. Hucka, S. M. Keating, M. König, C. Myers, D. Nickerson, and D. Waltemath. Specifications of standards in systems and synthetic biology: status and developments in 2020. *Journal of Integrative Bioinformatics*, 17(2-3):20200022, 2020. doi:10.1515/jib-2020-0022.

[425] K. Tiwari, S. Kananathan, M. G. Roberts, J. P. Meyer, M. U. Sharif Shohan, A. Xavier, M. Maire, A. Zyoud, J. Men, S. Ng, T. V. N. Nguyen, M. Glont, et al. Reproducibility in systems biology modelling. *Molecular Systems Biology*, 17(2):e9982, 2021. doi:10.15252/msb.20209982.

[426] F. G. Listl, J. Fischer, D. Beyer, and M. Weyrich. Knowledge Representation in Modeling and Simulation: A survey for the production and logistic domain. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1051–1056, Sept. 2020. doi:10.1109/ETFA46521.2020.9211994. URL `https://ieeexplore.ieee.org/abstract/document/9211994`. ISSN: 1946-0759.

[427] M. Schröder, S. Stählke, J. B. Nebe, and F. Krüger. Towards in-situ knowledge acquisition for research data provenance from electronic lab notebooks. In *DaMaLOS - First Workshop on Data and Research Objects Management for Linked Open Science, Co-located at the International Semantic Web Conference ISWC 2020, Virtual Conference*, pages 1–12, 2020. doi:10.4126/FRL01-006423288.

[428] F. Krüger and D. Schindler. A literature review on methods for the extraction of usage statements of software and data. *Computing in Science Engineering*, 22(1):26–38, 2020. doi:10.1109/MCSE.2019.2943847.

[429] D. Kerzel, B. König-Ries, and S. Sheeba. Mlprovlab: Provenance management for data science notebooks. In *BTW 2023*, pages 965–980. Gesellschaft für Informatik e.V., Bonn, 2023. ISBN 978-3-88579-725-8. doi:10.18420/BTW2023-66.

[430] A. Schreiber, L. von Kurnatowski, A. Meinecke, and C. de Boer. Visualization of Software Development Provenance. In H. Mori and Y. Asahi, editors, *Human Interface and the Management of Information*, pages 121–139, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-60114-9. doi:10.1007/978-3-031-60114-9_10.

[431] T. B. Noor and H. Hemmati. A similarity-based approach for test case prioritization using historical failure data. In *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, pages 58–68, 2015. doi:10.1109/ISSRE.2015.7381799.

[432] C. H. Koh, S. K. Palaniappan, P. Thiagarajan, and L. Wong. Improved statistical model checking methods for pathway analysis. In *BMC Bioinformatics*, volume 13, page S15. Springer, 2012. doi:10.1186/1471-2105-13-S17-S15.

[433] A. Davies, F. Hooley, P. Causey-Freeman, I. Eleftheriou, and G. Moulton. Using interactive digital notebooks for bioscience and informatics education. *PLOS Computational Biology*, 16(11):1–19, 2020. doi:10.1371/journal.pcbi.1008326.

[434] N. Feldkamp, S. Bergmann, and S. Strassburger. Knowledge discovery in simulation data. *ACM Transactions on Modeling and Computer Simulation*, 30(4), 2020. doi:10.1145/3391299.

[435] M. Zhang, S. Kim, P. Y. Lu, and M. Soljačić. Deep learning and symbolic regression for discovering parametric equations. *arXiv*, 2023. doi:10.48550/arXiv.2207.00529. Preprint.

[436] J. He, E. Bartocci, D. Ničković, H. Isakovic, and R. Grosu. DeepSTL: from english requirements to signal temporal logic. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, pages 610–622. ACM, 2022. doi:10.1145/3510003.3510171.

[437] D. J. Wagg, K. Worden, R. J. Barthorpe, and P. Gardner. Digital twins: State-of-the-art and future directions for modeling and simulation in engineering dynamics applications. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 6(3):030901, 2020. doi:10.1115/1.4046739.

[438] L. Crielaard, J. F. Uleman, B. D. Châtel, S. Epskamp, P. Sloot, and R. Quax. Refining the causal loop diagram: A tutorial for maximizing the contribution of domain expertise in computational system dynamics modeling. *Psychological methods*, 29(1):169–201, 2024. doi:doi.org/10.1037/met0000484.

[439] P. A. Bonatti, S. Decker, A. Polleres, and V. Presutti. Knowledge graphs: New directions for knowledge representation on the semantic web (Dagstuhl seminar 18371). *Dagstuhl Reports*, 8(9):29–111, 2019. doi:10.4230/DagRep.8.9.29. Publisher: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

[440] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu. BioGPT: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23(6):bbac409, 2022. doi:10.1093/bib/bbac409.

[441] W. M. van der Aalst. Process mining and simulation: a match made in heaven! In *Proceedings of the 50th Computer Simulation Conference*, SummerSim '18, pages 1–12, 2018. doi:10.5555/3275382.3275386.

[442] A. Macías, D. Muñoz, E. Navarro, and P. González. Data fabric and digital twins: An integrated approach for data fusion design and evaluation of pervasive systems. *Information Fusion*, 103:102139, 2024. doi:10.1016/j.inffus.2023.102139.

[443] A. Zela, A. Klein, S. Falkner, and F. Hutter. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. *arXiv*, 2018. doi:10.48550/arxiv.1807.06906. Preprint.

[444] S. R. Hunter, E. A. Applegate, V. Arora, B. Chong, K. Cooper, O. Rincón-Guevara, and C. Vivas-Valencia. An introduction to multiobjective simulation optimization. *ACM Transactions on Modeling and Computer Simulation*, 29(1), 2019. doi:10.1145/3299872.

[445] G. ten Broeke, G. van Voorn, and A. Ligtenberg. Which sensitivity analysis method should i use for my agent-based model? *Journal of Artificial Societies and Social Simulation*, 19 (1):5, 2016. doi:10.18564/jasss.2857.

[446] M. Horridge, N. Drummond, J. Goodwin, A. L. Rector, R. Stevens, and H. Wang. The Manchester OWL Syntax. In *OWLed*, volume 216, 2006. URL `https://ceur-ws.org/Vol-216/submission_9.pdf`. Last accessed 19 July, 2024.

[447] M. A. Musen. The Protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12, 2015. doi:10.1145/2757001.2757003.

[448] J. Herman and W. Usher. SALib: an open-source python library for sensitivity analysis. *Journal of Open Source Software*, 2(9):97, 2017. doi:10.21105/joss.00097.

[449] B. Iooss, A. Janon, G. Pujol, et al. *Sensitivity: Global Sensitivity Analysis of Model Outputs*, 2019. URL `https://CRAN.R-project.org/package=sensitivity`. R package version 1.16.1, last accessed 19 July, 2024.

[450] M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web Journal*, 2(1):11–21, 2011. URL `https://www.semantic-web-journal.net/content/owl-api-java-api-owl-ontologies`. Last accessed 19 July, 2024.

[451] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. HermiT: an OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014. doi:10.1007/s10817-014-9305-1.

[452] M. Funk, S. Hosemann, J. C. Jung, and C. Lutz. Towards ontology construction with language models. *arXiv*, 2023. doi:10.48550/arXiv.2309.09898. Preprint.