Guidelines for Reproducibly Building and Simulating Systems Biology Models

J. Kyle Medley*, Arthur P. Goldberg, and Jonathan R. Karr

Abstract*—*Objective: **Reproducibility is the cornerstone of the scientific method. However, currently, many systems biology models cannot easily be reproduced. This paper presents methods that address this problem.** ***Methods:*** **We analyzed the recent** ***Mycoplasma genitalium*** **whole-cell (WC) model to determine the requirements for reproducible modeling.** ***Results:*** **We determined that reproducible modeling requires both repeatable model building and repeatable simulation.** ***Conclusion:*** **New standards and simulation software tools are needed to enhance and verify the reproducibility of modeling. New standards are needed to explicitly document every data source and assumption, and new deterministic parallel simulation tools are needed to quickly simulate large, complex models.** ***Significance:*** **We anticipate that these new standards and software will enable researchers to reproducibly build and simulate more complex models, including WC models.**

***Index Terms*—Computational modeling, provenance, repeatability, reproducibility, standards, systems biology, whole-cell.**

I. Introduction

Reproducibility is one of the central tenets of the scientific method. We define *reproducibility* as the ability to confirm a result via a completely independent test, including different investigators, experimental methods, and experimental machinery. In the context of systems biology, a simulation result is *reproducible* if the model that generated the result can be recreated from our collective scientific knowledge, including manuscripts, databases, code repositories, and supplementary materials, and the simulation result can be regenerated from descriptions of the model and the simulation experiment. This rigorous standard eliminates conclusions that are based on incorrect methods, machinery, or experiments, and ensures that scientific results are only accepted as facts once multiple scientists have thoroughly dismissed other potential explanations.

We define *repeatability* as the ability to regenerate a result given the same experimental machinery and conditions. In the context of systems biology, a simulation result is *repeatable* if the numerical result can be regenerated from descriptions of the model and the simulation experiment. Repeatability is a more lenient standard than reproducibility because it does not require regeneration of the model itself. Consequently, testing repeatability only eliminates scientific conclusions that are based on erroneous experiments, and cannot eliminate conclusions that are based on faulty models.

To illustrate the distinction between reproducibility and repeatability, consider the following example: A modeler Alice wants to investigate a discrepant simulation result published by Bob. Bob's model predicts that knocking out regulator Y causes cancer, whereas Alice's model indicates that cancer requires at least two knockouts. Fortunately, Alice can investigate Bob's results because Bob published his model and simulation experiments in standard formats. Alice uses Bob's model and simulation experiment files to *repeat* Bob's simulation results, compares the two models, and finds that Bob's model generates different predictions because it uses a different rate law to describe the effect of regulator Y. However, because Bob's model file does not describe all of the experimental data and assumptions underlying his model, Alice cannot *reproduce* Bob's rate laws and thus cannot fully resolve why their models generate different predictions.

The systems biology community, spearheaded by the Computational Modeling in Biology Network [1], has developed several standard formats to exchange models and repeat simulations [2]. Examples of these standards include CellML [3], the COMBINE archive [4], the Systems Biology Markup Language (SBML) [5], the Simulation Experiment Description Markup Language (SED-ML) [6], and the Systems Biology Graphical Notation (SBGN) [7]. These standards support several types of models including ordinary differential equation (ODE), flux balance analysis (FBA) [8], and logical models. Numerous software programs support these standards [9]. Furthermore, several model repositories, including BioModels [10] and the CellML Model Repository [11], share models that are encoded in these standards. In addition, the bioinformatics and computer science communities have developed several methods and tools, including Galaxy [12], Taverna [13], and VisTrails [14], to track the provenance of computational analyses, including recording all of the data sources and assumptions used to conduct analyses.

These standards and modeling software tools help researchers repeat most systems biology simulation experiments and reuse, modify, expand, and combine most systems biology models. However, these standards and software tools provide limited support for regenerating models because they do not record all design choices, such as experimental data sources and assumptions, that are used to build models.

Manuscript received November 4, 2015; revised March 30, 2016 and June 7, 2016; accepted July 7, 2016. Date of publication July 18, 2016; date of current version September 16, 2016. J. K. Medley was supported by the National Institutes of Health under Grant R01 GM081070 and the National Science Foundation under Grant 1355909. A. P. Goldberg was supported by the Icahn Institute for Genomics and Multiscale Biology. J. R. Karr was supported by the National Science Foundation under Grant 1548123 and the James S. McDonnell Foundation Postdoctoral Fellowship Award in Studying Complex Systems. *Asterisk indicates corresponding author.*

*J. K. Medley is with the Department of Bioengineering, University of Washington, Seattle, WA 98195 USA (e-mail: medleyj@uw.edu).

A. P. Goldberg and J. R. Karr are with the Department of Genetics & Genomic Sciences, Icahn School of Medicine at Mount Sinai.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org (File size: 1.2 kB).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBME.2016.2591960

Furthermore, researchers have begun to develop more complex models that cannot be represented by the existing standards or simulated by the existing standard-compliant software. For example, several aspects of the recent whole-cell (WC) model of *Mycoplasma genitalium* [15] cannot currently be represented by SBML [16]. In particular, SBML cannot represent the multialgorithmic nature of the model. In addition, no SBML-compatible simulation software program supports all of the SBML packages needed to simulate WC models, including the Arrays [17], Distributions [18], Flux Balance Constraints [19], Hierarchical Model Composition [20], and Multistate and Multicomponent Species [21] packages.

Here, we present a path toward fully reproducible systems biology modeling, including WC modeling. First, we propose several requirements for reproducible modeling. Second, we outline several gaps in the existing standards and software for reproducible modeling and describe several new standards and software tools that are needed to achieve reproducible modeling. Third, we describe several methods for verifying repeatability. Achieving this vision will require significant research on standards and software development.

II. The Requirements for Reproducible Modeling

There are three requirements for fully reproducible systems biology modeling (see Fig. 1).

1) Researchers should be able to regenerate models, including every species and reaction, from the literature. Consequently, researchers should record the provenance of every data source and assumption used to build models, as well as save a copy of each data source to guarantee future access to every source.
2) Researchers should be able to regenerate statistically identical simulation results. Consequently, researchers should record every parameter value, algorithm, and simulation software option used to simulate models.
3) Researchers should ensure that multiple simulation software tools generate statistically identical results. This is particularly helpful for identifying errors in complex simulation software programs.

This requires standard model description formats that support all systems biology models, including WC models. This would enable researchers to use different software programs to generate the same results.

III. Toward a Platform for Reproducible Modeling

Currently, each individual modeler is responsible for reproducibly conducting their own research. This places a substantial burden on individual researchers to make every step of their research reproducible. To ease this burden, we recommend that the systems biology community develops several software tools, databases, and standards to enable researchers to conveniently conduct reproducible research.

1) More comprehensive experimental databases should be developed to provide the data needed to build systems models. For example, Karr *et al.* developed the WholeCellKB database [22] to organize more than 1400 quantitative measurements used to build the *M. genitalium* WC model. These databases should be machine-readable so that they can be automatically queried and used to build models. These databases should also use data models similar to that of systems biology models to clarify the connection between models and their underlying data.
2) New model design tools and Systems Biology Ontology (SBO) [23] terms are needed to record how models are built, including every design choice, assumption, and data source. Currently, researchers can use SBML annotations and SBO terms to record several common assumptions, such as the rapid equilibrium between free enzymes and enzyme-substrate complexes in Michaelis–Menten kinetics. However, many researchers do not utilize these annotations and the SBO does not represent all possible assumptions. Thus, researchers should develop model design tools that automatically record assumptions and data sources by adopting provenance tracking techniques [12]–[14]. The SBO should also be expanded to represent more assumptions.
3) Where possible, researchers should use standard formats such as SBML and SED-ML rather than proprietary codes to describe models and simulation experiments. This should include annotation of the units of every parameter.
4) The existing standard model description formats should be expanded to represent all types of systems biology models, including WC models. For example, to describe WC models in a standard format, SBML could be expanded to support genomic sequence data and sequence-based reaction patterns or SBML could be linked with the Synthetic Biology Open Language (SBOL) [24], which is already capable of representing sequence data.
5) The existing standard-compliant simulation software programs should be expanded to support a wider variety of models. In particular, simulation software programs should be expanded to support the Arrays, Distributions, Flux Balance Constraints, Hierarchical Model Composition, and Multistate and Multicomponent Species SBML packages. Several simulation software programs, including BioUML [25] and iBioSim [26], have already begun to support these packages. Unfortunately, most simulation software developers have insufficient funding to implement every package.
6) Modeling workflows should be expanded to systematically verify the statistical repeatability of simulation results.
7) New model verification tools should be developed to automatically identify errors in models. We developed a suite of tests to systematically error-check our *M. genitalium* WC model.[1] These tests checked for simple problems such as undefined species, reaction mass and charge imbalance, and inconsistent reaction rates among submodels. Nevertheless, we found this test suite invaluable for debugging our model. These tests should be generalized and new

[1] https://github.com/CovertLab/WholeCell/tree/master/src_test

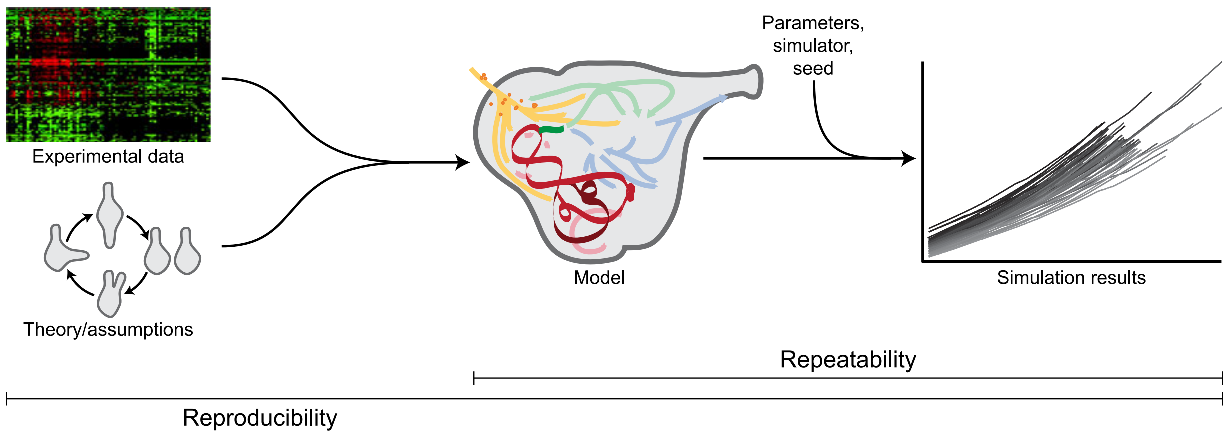

Fig. 1. Three requirements for reproducible systems biology modeling. 1) Provenance of every data source and assumption should be recorded so that models can be regenerated, including every species, reaction, and rate law. 2) Every simulation parameter value should be recorded so that numerical simulation results can be regenerated. 3) Multiple simulation software tools should generate statistically identical numerical results. Reproducibility is a stricter standard than repeatability. Models are considered repeatable, but not reproducible, if they satisfy just the second and third criteria.

software tools should be developed to help researchers systematically evaluate such tests.

8) Every model, simulation experiment, simulation result, and simulation software program should be reusable, extensible, documented, and published open-source [27].
9) Journals should require and help authors permanently archive every reported data source and model.

A. *Special Considerations for Stochastic Simulation*

Stochastic simulations typically use pseudorandom number generators (PRNGs) that deterministically produce the same numbers when seeded with the same initial state. Therefore, we recommend that stochastic simulation software developers provide ways to set and record PRNG seeds so that modelers can repeat stochastic simulation results. This would enable researchers to repeat not only statistical distributions, but exact trajectories, which is invaluable for identifying and debugging errors in complex models and simulation algorithms.

Furthermore, different implementations of the same PRNG often differ in subtle ways. For example, the seed methods of the Boost C++ Library [28] and Python Standard Library implementations of the Mersenne–Twister algorithm differentially initialize the PRNGs' internal states. Thus, we recommend the creation of standard PRNGs, including documentation of their algorithms, parameterizations, and initial states, and a test suite to help software developers verify their implementations.

B. *Special Considerations for Multialgorithm WC Models*

Multialgorithm WC models strive to represent every gene and cell function by combining multiple submodels of individual cellular pathways, each represented using different mathematics and each trained using different experimental data [29]–[31]. For example, the *M. genitalium* WC model was composed of 28 submodels, including an FBA submodel that describes its metabolism, stochastic submodels that describe its transcription and translation, and an ODE submodel that describes its division. This multialgorithm methodology is motivated by the desire to model biological systems as completely as possible, given our current knowledge, by simultaneously using fine-grained representations and coarse-grained representations to represent well- and poorly characterized pathways, respectively.

Multialgorithm modeling is a new methodology that still lacks a rigorous theoretical foundation. Consequently, significant work is needed to develop tools for building, simulating, and reproducing WC models.

1) The Hierarchical Model Composition SBML package should be extended to use the KiSAO ontology [32] to represent each submodel's simulation algorithm.
2) Researchers should determine how to concurrently integrate submodels that share state. Researchers are currently exploring several potential methods to concurrently integrate submodels, including shared memory and parallel discrete event simulation [33].
3) Researchers should develop a high-performance, reusable multialgorithm simulator.
4) Additional tools should be developed to help researchers build and analyze WC models.
5) These programs should be implemented as separate tools and integrated into a comprehensive WC modeling platform so that software developers can contribute to individual tools and so that modelers can easily use alternative components.

We anticipate that this platform will enable more researchers to engage in WC modeling and accelerate the WC modeling field.

C. *Special Considerations for Parallel Simulation Software Programs*

Multithreaded and distributed computing are two important methods for accelerating the simulation of computationally expensive models such as WC models. Parallel programs use multiple threads and/or cores to simultaneously execute different parts of a simulation. The order in which these threads and/or cores complete computations depends on the operating environment, and the order in which computations complete can affect subsequent computations and, in turn, simulation results. Other sources of nondeterminism in parallel computing include memory layout, delays caused by devices and processes, and user input. Thus, parallel programs can generate nondeterministic simulation results due to variable operating environments.

For example, if two WC submodels that are executed in parallel attempt to bind proteins to the same site on the chromosome, the submodel that completes its computations first will bind the protein to the chromosome. However, when the simulation is rerun under a different load on the computer, the other submodel may complete its computations first, resulting in the binding of a different protein to the site. This small difference can lead to further differences between later time steps, resulting in different simulation results solely due to different operating environments.

Consequently, special care must be taken to create deterministic parallel simulation programs. Therefore, we briefly examine methods from computer science that have been developed to create deterministic parallel programs.

Several libraries have been developed to eliminate nondeterminism from multithreaded programs. These include COREDET [35] and DTHREADS [36]. COREDET is a library for compiling C/C++ programs such that they behave deterministically. DTHREADS is a replacement for the standard UNIX Pthreads library. Both systems create deterministic programs by preventing information sharing between threads during parallel phases, and deterministically allowing sharing during serial phases. These systems also eliminate nondeterminism due to variable memory layout. DTHREADS achieves this by implementing a special deterministic memory allocator. COREDET deterministically allocates memory by compiling memory allocators with the COREDET library to generate deterministic memory allocators. Both COREDET and DTHREADS have been shown to impose minimal overhead [36]. We recommend that simulation software developers implement options for deterministic, repeatable simulations, and we offer COREDET and DTHREADS as examples of how this can be achieved for multithreaded simulation programs.

Computer scientists have also developed several practices to create deterministic distributed programs. First, interprocess communication messages should be deterministically executed to avoid race conditions [37]. Messages can be deterministically scheduled by using first-in-first-out message queues between each pair of communicating processes and blocking message receive operations [38]. Time-driven simulations can also deterministically schedule messages by ordering messages based on their simulation times [39]. Parallel shared-memory time step simulations can be made deterministic by synchronizing the end of time steps with barriers [40], such as implemented in OpenMP [41], and employing deterministic approaches to share time step state updates among threads. Second, each individual process in a distributed simulation program should be deterministic. This can be achieved by following the recommendations in this paper. We recommend that simulation software developers adopt these practices to create deterministic distributed simulation programs.

Last, we recognize that it is challenging and time-consuming to implement deterministic parallel software. Thus, we recommend that each software developer evaluate the costs and benefits of implementing deterministic programs. However, we hope that more developers will choose to implement deterministic software, particularly as complex hybrid models, which need deterministic simulations for debugging and verification, become more popular.

IV. Methods for Verifying Repeatability

After adopting the methods outlined above to repeatably simulate models, we recommend that systems biology modelers and software developers verify repeatability by testing the equivalence of simulation results generated by multiple simulation software programs for the same model, and then correcting any discrepancies identified by this analysis. Bergmann and Sauro [42] and Evans *et al.* [43] have used this approach to assess the repeatability of models across several simulators, identifying errors in several simulation programs.

Model checking and unit testing are two formalisms that can be used to verify repeatability. Simulation-based model checkers, such as SPIN [44], are extensively used in electrical engineering to verify the behavior of models. Simulation-based checkers execute multiple simulations and then verify that the results are consistent with the user-supplied specification of the model's behavior [45]. Model verification systems should be adopted to verify the repeatability of systems biology models. Unit testing is a powerful strategy for verifying the behavior of software. Unit testing could also be used to verify the repeatability of systems biology models and simulation programs by using multiple simulators to execute multiple simulations and verifying that their predictions are statistically identical.

A. *Special Considerations for Stochastic Models*

The repeatability of stochastic models can be assessed by verifying the statistical equivalence of simulation results across multiple simulators. Statistical equivalence should be tested using multivariate Kolmogorov–Smirnov tests [46]. Tests that only check the predicted mean and variance are insufficient because most model predictions cannot be fully specified by their mean and variance. For example, the bistable system illustrated in Fig. 2 cannot be specified by its mean and variance.

However, it is often challenging to statistically verify the repeatability of complex algorithms and models across multiple simulators because large numbers of simulations are needed to statistically verify repeatability with high confidence, especially for models with rare events [47]. Instead, we recommend that researchers who are developing novel simulation algorithms and large-scale models utilize the methods described in the previous section to simulate models deterministically and verify that each simulation software program can exactly regenerate its own numerical results. Although this approach does not ensure repeatability across simulators, it does facilitate the debugging of complex simulation algorithms.

	Reaction	Rate
(1)	$\varnothing \to x$	$\frac{1}{2} + V_m \frac{x^n}{K_m + x^n}$
(2)	$x \to \varnothing$	$\frac{k}{x}$

Where

$x = 24\,\mu$M at $t = 0\,$s,

$n = 4,$

$V_m = 4\,s^{-1},$

$K_m = 10^6\,\mu$M, and

$k = 0.125\,\mu\text{M}\,\text{s}^{-1}.$

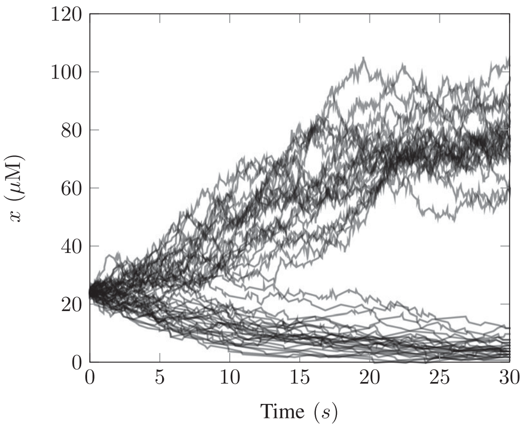

Fig. 2. Mean and variance testing should not be used to verify the repeatability of stochastic models because most simulation results cannot be uniquely specified by these statistical metrics. For example, the bistable stochastic model described in (a) and illustrated in (b) cannot be specified by its mean and variance. Statistical equivalence should be tested using multivariate Kolmogorov–Smirnov tests. The bistable stochastic model described in (a) includes two reactions. Reaction (1) represents the production of species x and reaction (2) represents the degradation of x. Supplementary Material S1 contains Python and Antimony [34] code for this example.

B. *Special Considerations for Hierarchical Models*

The repeatability of hierarchical models, such as WC models, can be efficiently verified by taking advantage of their hierarchical structure in the following two ways: 1) Modelers should verify the repeatability of each submodel. This also facilitates debugging by beginning with small, easily testable units. 2) Modelers should verify the repeatability of the combined model.

C. *Special Considerations for Multialgorithm WC Models*

As introduced above, multialgorithm modeling is a new methodology that still requires significant fundamental research. Thus, multialgorithm simulation algorithms will need to be extensively tested to verify their repeatability, as well as to understand their fundamental properties.

D. *Special Considerations for Chaotic Models*

Chaotic systems pose special challenges for repeatability and reproducibility. One criterion for judging whether a model is chaotic is the Lyapunov exponent, which measures the divergence between two trajectories starting at infinitesimally close initial conditions. A positive value indicates that the distance between the trajectories increases exponentially with time and implies that the model is chaotic. An extensive analysis of the reproducibility of chaotic systems is beyond the scope of this paper. However, we suggest three guidelines for repeating simulations of chaotic systems. First, Lyapunov exponents should be used to gauge whether a model is chaotic. Second, statistical tests of the type discussed in Section IV should not be used to evaluate the reproducibility of chaotic models. Third, simulation software developers should use high precision to encode initial conditions and parameters so that chaotic model simulation results can be repeated.

V. Conclusion

Substantial work is needed to address the numerous challenges to enhancing the reproducibility of systems biology modeling, including developing new standards and simulation software tools. These standards and software tools should enable researchers to regenerate models from our scientific knowledge by recording the provenance of every model design decision, assumption, data source, parameter, and software option. These software tools should also enable researchers to regenerate simulation results by using PRNGs and deterministic multithreading libraries. This requires expanded standards and software tools that support all systems biology models, including WC models. These simulation software tools should also be extensively tested to verify that they produce consistent simulation results. In turn, this requires a strong commitment among the scientific community to high-quality, open-source software development, including more emphasis on the publication of software programs in academic journals.

Acknowledgment

The authors thank G. Gossel, P. Mendes, and H. Sauro for thoughtful discussions on reproducibility. The content is solely the responsibility of the authors and does not necessarily represent the views of the National Institutes of Health, the National Science Foundation, or the James S. McDonnell Foundation.

References

[1] M. Hucka *et al.*, “Promoting coordinated development of community-based information standards for modeling in biology: The COMBINE initiative,” *Frontiers Bioeng. Biotechnol.*, vol. 3, 2015.

[2] A. Dräger and B. Ø. Palsson, “Improving collaboration by standardization efforts in systems biology,” *Frontiers Bioeng. Biotechnol.*, vol. 2, 2014.

[3] A. A. Cuellar *et al.*, “An overview of CellML 1.1, a biological model description language,” *Simulation*, vol. 79, no. 12, pp. 740–747, 2003.

[4] COMBINE. COMBINE. 2012. [Online]. Available: http://co.mbine.org/events/COMBINE_2012

[5] M. Hucka *et al.*, "The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524–531, 2003.
[6] D. Waltemath *et al.*, "Reproducible computational biology experiments with SED-ML—The simulation experiment description markup language," *BMC Syst. Biol.*, vol. 5, no. 1, p. 198, 2011.
[7] N. Le Novère *et al.*, "The systems biology graphical notation," *Nature Biotechnol.*, vol. 27, pp. 735–741, 2009.
[8] J. D. Orth *et al.*, "What is flux balance analysis?," *Nature Biotechnol.*, vol. 28, no. 3, pp. 245–248, 2010.
[9] M. Hucka *et al.*, "A profile of today's SBML-compatible software," in *Proc. IEEE 7th Int. Conf. e-Sci. Workshops*, 2011, pp. 143–150.
[10] V. Chelliah *et al.*, "BioModels: Ten-year anniversary," *Nucleic Acids Res.*, vol. 43, no. D1, pp. D542–D548, 2015.
[11] C. M. Lloyd *et al.*, "The CellML model repository," *Bioinformatics*, vol. 24, no. 18, pp. 2122–2123, 2008.
[12] J. Hillman-Jackson *et al.*, "Using galaxy to perform large-scale interactive data analyses," *Current Protocols Bioinf.*, 2012, pp. 10–5, ISBN: 9780471250951.
[13] T. Oinn *et al.*, "Taverna: A tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, 2004.
[14] S. P. Callahan *et al.*, "Vistrails: Visualization meets data management," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2006, pp. 745–747.
[15] J. R. Karr *et al.*, "A whole-cell computational model predicts phenotype from genotype," *Cell*, vol. 150, no. 2, pp. 389–401, 2012.
[16] D. Waltemath *et al.*, "Toward community standards and software for whole-cell modeling," *IEEE Trans. Biomed. Engr.*, 2016, to be published.
[17] L. Watanabe and C. J. Myers, "Efficient analysis of SBML models of cellular populations using arrays," *ACS Synthesis Biol.*, doi:10.1021/acssynbio.5b00242.
[18] S. L. Moodie *et al.*, "The distributions package for SBML level 3," 2015, Accessed on : Feb. 02, 2016. [Online]. Available: http://sourceforge.net/p/sbml/code/HEAD/tree/trunk/specifications/sbml- level-3/version-1/distrib/sbml-level-3-distrib-package-proposal.pdf?format=raw
[19] B. G. Olivier and F. T. Bergmann, "The systems biology markup language (SBML) level 3 package: Flux balance constraints," *J. Integr. Bioinform.*, vol. 12, no. 2, p. 269, 2015.
[20] L. P. Smith *et al.*, "SBML Level 3 package: Hierarchical model composition, version 1 release 3," *J. Integr. Bioinform.*, vol. 12, no. 2, p. 268, 2015.
[21] F. Zhang and M. Meier-Schellersheim, "SBML Level 3 package specification: Multistate/Multicomponent Species (Version 1, Release 0.1 Draft 369)," 2015, Accessed on: May. 25, 2015. [Online]. Available: http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/multi
[22] J. R. Karr *et al.*, "WholeCellKB: Model organism databases for comprehensive whole-cell models," *Nucl. Acids Res.*, vol. 41, no. D1, pp. D787–D792, 2013.
[23] N. Juty and N. le Novère, "Systems biology ontology," *Enc. Syst. Biol.*, 2013, pp. 2063–2063, ISBN: 9781441998620.
[24] M. Galdzicki *et al.*, "The synthetic biology open language (SBOL) provides a community standard for communicating designs in synthetic biology," *Nature Biotechnol.*, vol. 32, no. 6, pp. 545–550, 2014.
[25] F. Kolpakov, "BioUML: Visual modeling, automated code generation and simulation of biological systems," in *Proc. BGRS*, 2006, vol. 3, pp. 281–285.
[26] J. T. Stevens and C. J. Myers, "Dynamic modeling of cellular populations within iBioSim," *ACS Synthesis Biol.*, vol. 2, no. 5, pp. 223–229, 2013.
[27] S. M. Easterbrook, "Open code for open science?," *Nature Geosci.*, vol. 7, no. 11, pp. 779–781, 2014.
[28] B. Schäling, *The boost C++ libraries*. 2nd ed. Boris Schäling: XML Press, Sep. 22, 2014.
[29] J. R. Karr *et al.*, "The principles of whole-cell modeling," *Current Opinion Microbiol.*, vol. 27, pp. 18–24, 2015.
[30] D. N. Macklin *et al.*, "The future of whole-cell modeling," *Current Opinion Biotechnol.*, vol. 28, pp. 111–115, 2014.
[31] J. Carrera and M. W. Covert, "Why build whole-cell models?," *Trends Cell Biol.*, vol. 25, no. 12, pp. 719–722, 2015.
[32] M. Courtot *et al.*, "Controlled vocabularies and semantics in systems biology," *Mol. Syst. Biol.*, vol. 7, no. 1, p. 543, 2011.
[33] A. P. Goldberg *et al.*, "Toward scalable whole-cell modeling of human cells," in *Proc. Annu. ACM Conf. SIGSIM Principles Adv. Discrete Simul.*, 2016, pp. 259–262.
[34] L. P. Smith *et al.*, "Antimony: A modular model definition language," *Bioinformatics*, vol. 25, no. 18, pp. 2452–2454, 2009.
[35] T. Bergan *et al.*, "CoreDet: A compiler and runtime system for deterministic multithreaded execution," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 1, pp. 53–64, 2010.
[36] T. Liu *et al.*, "Dthreads: Efficient deterministic multithreading," in *Proc 23rd ACM Symp. Oper. Syst. Principles*, 2011, pp. 327–336.
[37] R. H. Netzer and B. P. Miller, "What are race conditions?: Some issues and formalizations," *ACM Lett. Program. Lang. Syst.*, vol. 1, no. 1, pp. 74–88, 1992.
[38] K. M. Chandy and J. Misra, "Asynchronous distributed simulation via a sequence of parallel computations," *Commun. ACM*, vol. 24, no. 4, pp. 198–206, 1981.
[39] D. R. Jefferson, "Virtual time," *ACM Trans. Program. Lang. Syst.*, vol. 7, no. 3, pp. 404–425, 1985.
[40] E. D. Brooks III, "The butterfly barrier," *Int. J. Parallel Program.*, vol. 15, no. 4, pp. 295–307, 1986.
[41] L. Dagum and R. Enon, "Openmp: An industry standard API for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan. 1998.
[42] F. T. Bergmann and H. M. Sauro, "Comparing simulation results of SBML capable simulators," *Bioinformatics*, vol. 24, no. 17, pp. 1963–1965, 2008.
[43] T. W. Evans *et al.*, "The SBML discrete stochastic models test suite," *Bioinformatics*, vol. 24, no. 2, pp. 285–286, 2008.
[44] G. J. Holzmann, "The model checker SPIN," *IEEE Trans. Softw. Eng.*, vol. 23, no. 5, pp. 279–295, May 1997.
[45] M. Kwiatkowska *et al.*, "Prism 4.0: Verification of probabilistic real-time systems," in *Proc. Comput. Aided Verification*, 2011, pp. 585–591.
[46] A. Justel *et al.*, "A multivariate Kolmogorov-Smirnov test of goodness of fit," *Statist. Probab. Lett.*, vol. 35, no. 3, pp. 251–259, 1997.
[47] K. H. Kim *et al.*, "Nonlinear biochemical signal processing via noise propagation," *J. Chem. Phys.*, vol. 139, no. 14, 2013, Art. no. 144108.

J. Kyle Medley received the BSc degree in mechanical engineering from the University of Missouri-Kansas City, Kansas City, MO, USA in 2014 and is currently workingt toward the Ph.D. degree in bioengineering from the University of Washington, Seattle, WA, USA.

His research interests include modeling dynamical biophysical processes such as metabolism and gene expression regulation. He currently develops libRoadRunner, a software package for simulating dynamical models encoded in SBML.

Arthur P. Goldberg received the Ph.D. degree in computer science from the University of California, Los Angeles, Los Angeles, CA, USA, in 1991 and the A.B. degree in astronomy and astrophysics from Harvard College, Cambridge, MA, USA, in 1977.

He is an Associate Professor at the Icahn School of Medicine at Mount Sinai, New York, NY, USA. His research interest includes the development of high-performance simulation technologies for WC modeling.

Jonathan R. Karr received the B.S.s degree in physics and brain & cognitive sciences from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2006, the M.S. degree in medicine from Stanford University, Stanford, CA, USA, in 2014, and the Ph.D. degree in biophysics in 2014.

He is currently a Fellow at the Icahn School of Medicine at Mount Sinai, New York, NY, USA. His research interest includes the development of comprehensive WC computational models and their application to bioengineering and medicine.