A Profile of Today's SBML-Compatible Software

Michael Hucka*, Frank T. Bergmann*, Sarah M. Keating*†, Lucian P. Smith‡
*California Institute of Technology, Pasadena, CA, USA
†EMBL European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, UK
‡University of Washington, Seattle, WA, USA

Email: {mhucka, fbergman, skeating}@caltech.edu, lpsmith@u.washington.edu

***Abstract*—Computational systems biologists today have a healthy selection of software resources to help them do research. Many software packages, especially those concerned with computational modeling, have adopted SBML (the Systems Biology Markup Language) as a machine-readable format to permit users to exchange models. Our group has a keen interest in understanding the landscape of SBML support. To help us ascertain the state of modern SBML-compatible software, in mid-2011 we initiated a survey of software packages that support SBML. Here we report the preliminary survey results. Based on 81 packages for which we have data so far, we summarize the trends in six areas: (1) What are the major types of functionality offered by the software systems? (2) What mathematical frameworks do they support? (3) What are their SBML-specific capabilities? (4) What other standards do they support besides SBML? (5) What are their characteristics with respect to run-time environments? And finally, (6) what are the availability and licensing terms?**

***Index Terms*—systems biology; bioinformatics; computational systems biology; data handling; application software; open source software; software tools; software standards**

I. Introduction

Modern software systems are indispensable to research, and they are so deeply coupled to continued progress that there is a positive feedback loop at work: better and more sophisticated tools enable better and deeper research, which in turn lead to demand for new and more powerful software.

While software tools evolve, the products of research performed with them must survive for longer periods of time so that researchers everywhere may build upon each other's work. This in turn drives a need to agree on standard electronic formats for data of all kinds—not only experimental data, but the computational models, procedures, and results obtained with them. SBML, the Systems Biology Markup Language [1]–[3], has become a de facto standard for representing computational models in this field. By our count, at least 230 software systems have implemented SBML support over the past decade.

This bounty of software often leads to questions from users, such as, which tool(s) should they use for a particular task? Software developers also have questions; for example, they often want to know whether other tools implement capabilities similar to those of their own systems. Unfortunately, these and similar questions rarely have simple answers because there are so many dimensions to software systems. We have attempted to help people answer the questions for themselves by providing information resources on the SBML.org portal website, notably in the form of the *SBML Software Guide* [4]. Among the features of the Guide are (i) a table listing known SBML-compatible software tools along with their characteristics on particular dimensions, and (ii) a separate page giving narrative descriptions of different tools grouped into categories. However, with the rapid growth in the number and variety of SBML software, and the emergence of new related standards such as SBGN [5] and SED-ML [6], it has become clear that better and more detailed information resources are needed.

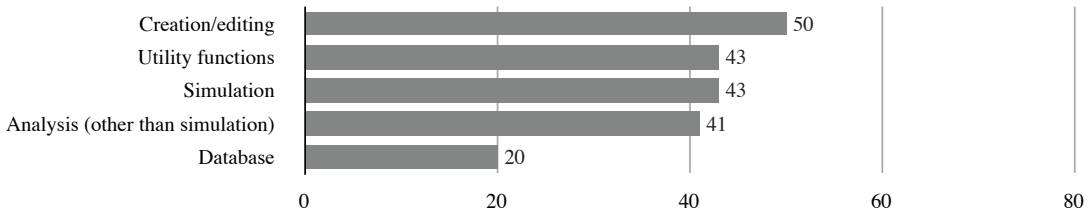
For the benefit of the systems biology community, as well as to help us understand the state of SBML today, we initiated a project to overhaul the SBML Software Guide. As part of that effort, in May, 2011, we issued a new SBML software survey. Here we report on some of our results to date. We begin with an explanation of our methods in Section II. We report the raw results in Section III and provide interpretations of the results in Section IV. We close with some conclusions in Section V.

II. Methods

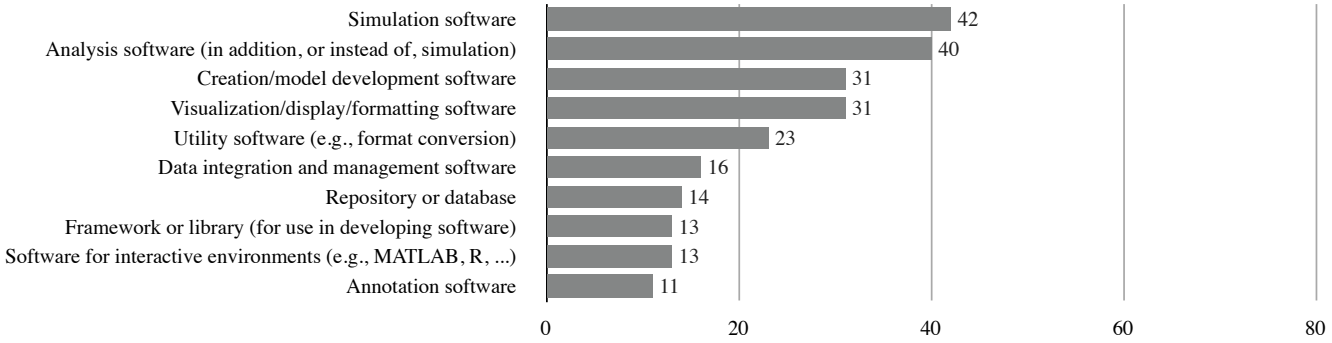
The *SBML Software Details Questionnaire* [7] is an electronic survey targeted at developers of SBML-compatible software systems. It contains a total of 28 questions. We implemented the survey using a commercial Web-based service [8].

We used multiple approaches to notify developers and request that they complete the questionnaire. We announced the survey on several SBML-oriented mailing lists and to sysbio@caltech.edu, a general systems biology list. We made the first call for participation in May, 2011, and made a second call a few weeks later. We also sent direct email to the majority of the contact addresses associated with the software listed in the existing SBML Software Guide [4].

These approaches have produced a total of 85 responses to date. Of these, four were clearly incomplete (due to missing data), and therefore we omit them from the following analyses. This left a total of 81 usable responses for this article. We performed some additional due diligence on these remaining entries by comparing the responses to information given in documentation on the software's own web pages. In a few cases, we made small adjustments to the responses, in particular to add missing software dependencies or to note other standards supported by the software but not reported in the survey. We did not perform wholesale correction of the survey entries submitted by other people, because we are concerned this could just as easily introduce errors as correct them.



(a) Facilities provided directly by the software tools (without invoking other software). Bars indicate the total number of tools providing each facility.



(b) Categories of software. Bars indicate the total number of tools for which respondents judged the given category to apply.

Fig. 1. Survey responses for the two questions discussed in Section III-A. The total number of tools represented is 81. For each question, the response choices are not mutually exclusive, and consequently the summed number of responses can add up to more than 81.

III. Results

We begin by noting that the total number of software tools represented by the returns obtained so far (85) is significantly less than the 230 listed in today's SBML Software Guide [4]. We attribute the discrepancy to a combination of factors. First, the survey so far has only been available for less than three months, and it is likely that not all developers have taken the time to fill it out yet. (By contrast, the current SBML Software Guide has been active and collecting answers for many years.) Second, some of the software systems listed in the current Guide no longer appear to be actively maintained, but it is not easy to verify this in all cases. Part of our motivation for performing the current survey is to help identify these tools.

In this paper, we focus on a subset of the survey questions. We explain the questions and report the results below.

A. Capabilities of SBML-compatible software systems

SBML began life a decade ago as a format specifically intended to enable the direct exchange of models between different simulation packages. Since that time, the kinds of resources available for systems biology have increased considerably in variety and number. Consequently, one of the first topics we sought to explore is, what do today's SBML-compatible software systems offer in terms of main functionality? We probed this topic using two related questions.

In one of the two questions, we presented a short list of categories intended to cover the most fundamental facilities that we believe are provided by different software tools in this area. We then asked, "*What facilities does the package provide by itself (i.e., without invoking another package) for working with SBML? 'Creation' = creating/editing models, 'Simulation' = performing time-series simulation of models, 'Analysis' = analyzing models (e.g., sensitivity analysis, flux-balance analysis, etc.), 'Database' = implementing a database of models (and not merely access to an external database), and 'Utility' = providing other utility functions (e.g., translating SBML to/from other formats). Check all that apply.*" We summarize the results in Fig. 1(a).

In the second question, we provided a larger list of software categories and asked "*Which of the following categories best describe your software? (Check all that apply.)*". This larger list was created based on our own experiences with the current SBML Software Guide [4]. The list of options is shown, along with the survey results, in Fig. 1(b).

Although both questions deal with the capabilities of the tools, they differ in that the first is concerned with core functionality that a tool *implements*, while the second is concerned with what are the *purposes* at which the tool is aimed. Most tools offer a combination of capabilities and can be applied to a variety of purposes.

B. Mathematical frameworks in use

Related to the topic of the previous section is the following question: what kinds of mathematical frameworks are supported by these software systems? SBML itself is compatible with many frameworks, but different modelers and model designs find different frameworks suitable. What do today's software tools offer?

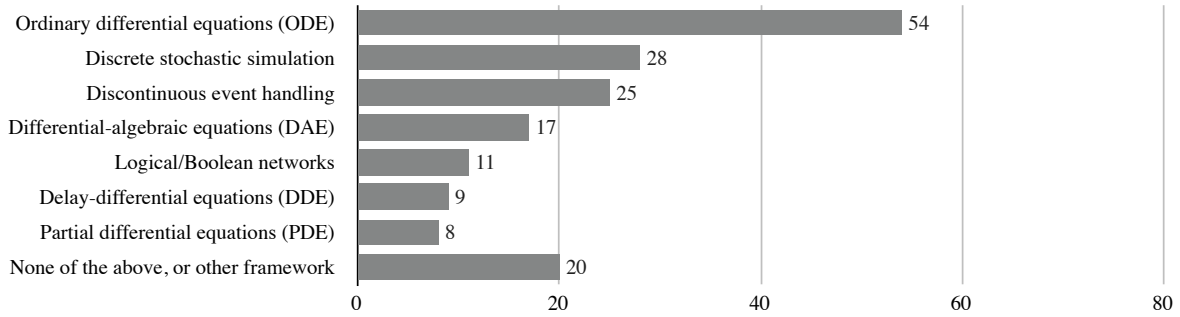


Fig. 2. Responses to the question discussed in Section III-B: mathematical frameworks supported by the software tools. Choices are not mutually exclusive.

We addressed this by listing a set of common frameworks and then asking the question, "*Regardless of whether your software provides simulation capabilities, what modeling frameworks does the package support when working with SBML files? (Check all that apply.)*" Fig. 2 lists the options we provided, and the responses to this question. Note that the options are not mutually exclusive; many systems support more than one framework. For the choice of "None of the above, or other", we also requested an explanation and provided a text box for the reply; we will return to this in Section IV-B.

C. SBML-specific software capabilities

Our survey included several questions related to the components of SBML understood by software packages. Many of the questions are of such a deeply SBML-specific nature that they are outside the scope of this paper, but we believe a few results may be of sufficiently general interest.

The first question is very basic: does the software import SBML, export SBML, or both? We report the results in Fig. 3.

Fig. 3. Proportion of tools that read SBML, write SBML, or both.

The second question concerns the editions of SBML that the software packages can interpret. Major editions of SBML are termed *Levels* and represent substantial changes to the composition and structure of the language. Minor revisions of SBML are termed *Versions* and constitute changes within a Level to correct, adjust, and refine language features. In practice, once a new Level of SBML is defined, no further development is undertaken on Versions of lower Levels. The most recent Level of SBML is Level 3, the first version of which was finalized in October, 2010 [9]. We issued an updated version of libSBML [10], an open-source library for working with SBML, at the same time as the release of the SBML Level 3 specification so that software developers could immediately begin working on Level 3 support for their tools. We are therefore very interested in assessing the degree of support of SBML Level 3 since its introduction.

Towards that goal, we asked respondents to indicate the highest Level/Version of SBML that their software could understand. Here we focus on the subset of results concerning support for SBML Level 3. The results are shown in Fig. 4.

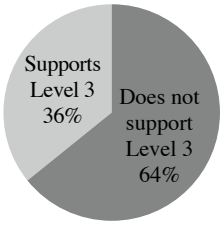


Fig. 4. Proportion of tools that support SBML Level 3.

A third question focused on the model-understanding capabilities of the software. Specifically, we asked "*In terms of interpretation/simulation/analysis/output performed by your software, what is the most complex degree of quantitative or mathematical understanding that your software can do with SBML models? Please pick all that are relevant.*" We provided a list of options that ranged from basic to relatively advanced, along with an "Other or not applicable" choice.

The choices offered in this question merit elaboration. The most basic degree of understanding is to display or manipulate species, reactions, parameters and/or compartments in an SBML model; this requires no simulation or dynamical analysis and can be performed by (e.g.) network visualization tools. A step up in degree of understanding is the ability to interpret stoichiometric relationships; this is typically needed for working with metabolic maps (such as those created in network reconstructions [11]) where the models lack rate laws. The next degree up from *that* is interpretation of reaction rate laws and reaction kinetics, and this is the bread and butter of many simulation tools in systems biology. Beyond this lie more specialized capabilities: handling other mathematical relationships besides those expressed by reactions and stoichiometries, handling discontinuous events (which requires hybrid numer-

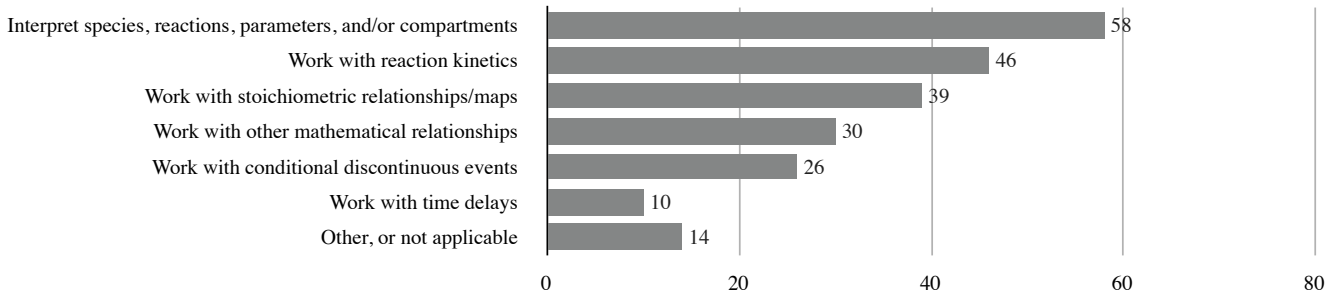


Fig. 5. Responses to the question discussed in Section III-C, regarding the software systems' quantitative or mathematical understanding of SBML models. Once again, choices are not mutually exclusive. Also see the important discussion of these results in Section IV-C and a revised graph in Fig. 9.

ical solvers), and/or handling time-delay expressions (which requires even more specialized numerical solvers).

We list the results of the survey in Fig. 5. Note that, again, the choices are not mutually exclusive, and many respondents selected more than one option for their software.

D. Other standards supported

SBML is not the only standard or proposed standard used in systems biology. We wondered what other, related standards are supported by SBML-compatible software packages. We asked respondents to provide a comma-separated list of each one supported by their software. We counted the times each different standard was mentioned and discounted those that are unrelated to systems biology (e.g., GraphML [12]). In Fig. 6, we show a graph of the frequency with which the remaining standards were mentioned in the survey results.

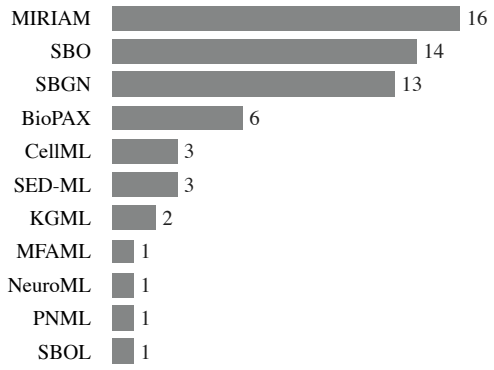


Fig. 6. Other standards or widely-used formats supported by tools reported in this survey. Bars show the number of times each standard/format was listed.

The standards that respondents mentioned are: MIRIAM (Minimum Information Requested in the Annotation of Models [13]), SBO (Systems Biology Ontology [14]), SBGN (Systems Biology Graphical Notation [5]), BioPAX (Biological Pathway Exchange [15]), CellML [16], SED-ML (Simulation Experiment Description Markup Language [6]), KGML [17], NeuroML [18], MFAML (Metabolic Flux Analysis Markup Language [19]), PNML (Petri Net Markup Language [20]), and SBOL (Synthetic Biology Open Language [21]).

E. General properties of the software implementations

Software portability and interoperability issues frequently can be the deciding factor in choosing one software package over another. Users need to know basic properties such as the operating systems on which SBML-compatible tools can run. We sought answers to this and other questions in the survey.

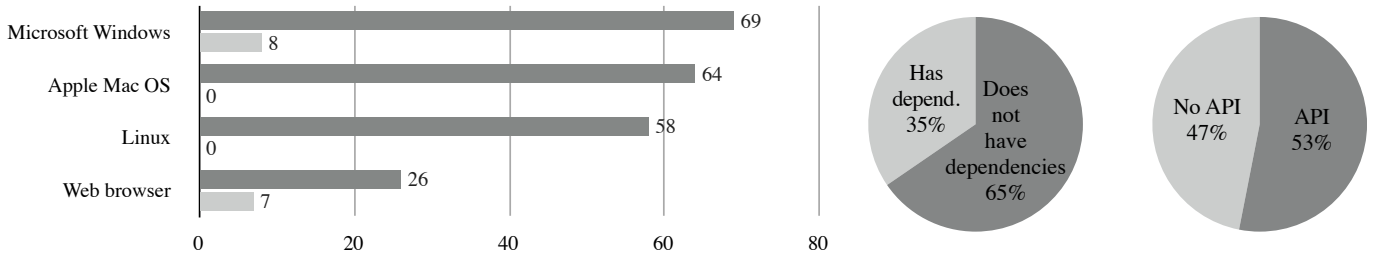
First, we asked respondents to list all the operating systems under which their software could run. We included the three dominant platforms (Windows, Linux and Mac OS) and "web browser" as four explicit options, and additionally, allowed for write-in answers in an "Other" field. Here we focus on only the main choices and show the results in Fig. 7(a).

We also asked whether the software tool depends on *other* software or environments, such that it is fundamentally necessary to have this other software or environment to use either the tool or its output. Examples of such environments are MATLAB [22] and R [23]. We gave special instructions with respect to scripting environments and converters. For scripting environments, we stated that the crucial point is whether the user interacts with the software by typing commands to the script language interpreter. If the language is merely used to implement the software tool, and users interact with it via a graphical user interface or other non-script-interface means, then it should not be counted. For converters, we gave the following instructions. If a converter targets *another* system, even if it does not require the system to be present when running, it should list the target system as a dependency. The logic behind this is that the output of the converter is not sufficiently useful to a user without the target system. The results of this question are shown in Fig. 7(b).

Finally, we asked whether the software tools expose an application programming interface (API), such that a user could program with it. This includes APIs for environments such as MATLAB [22], as well as web services and other network APIs. The results are presented in Fig. 7(c).

F. Availability and licensing

A final set of questions in the survey concerned the distribution and licensing terms of the software. The legal and practical consequences of different alternatives in this area can have a significant impact on users' choice of the tools they use.



(a) Major operating systems/platforms supported. (b) Dependencies. (c) APIs.

Fig. 7. (a) Number of applications that run on the most common operating systems and platforms. Dark gray (longer) bars indicate totals for each platform; light gray (shorter) bars indicate the subset of those tools that support *only* that platform. (b) Proportion of tools that depend on other environments such as MATLAB [22]. (c) Proportion of tools that provide an application programming interface (API).

We first asked respondents whether their software is available for free, or at cost, to academic and (separately) non-academic users. Fig. 8(a) and (b) summarize the results as percentages of the total responses.

We also asked whether the source code for the software was available. We specifically asked that this question be considered separately from whatever might be the *terms of redistribution* for the software source code. We show the results for this question in Fig. 8(c). (We did also include a separate question in the survey asking respondents to indicate whether redistribution was permitted under open-source terms. However, upon analyzing the results for that question, we now believe many respondents were confused about the intended meaning. We therefore omit that question's results entirely. We believe the other questions were not affected as a result of possible confusion over the redistribution question.)

Finally, we asked about licensing terms. We provided a free-form text field in the survey, for respondents to write the name of the license used (if it is a common license) or a web address detailing the terms. We tabulated the number of times different licenses were mentioned and graphed the results in Fig. 8(d).

IV. Discussion

In this section, we discuss the results reported above. We cover each topic in the same order as in Section III.

A. Capabilities of SBML-compatible software systems

Perhaps the most striking aspect of the results shown in Fig. 1 is how *many* software tools are available with each of the capabilities listed. For example, over half of the tools provide simulation, and well over half provide some kind of model creation/editing facility. When we consider that the number of responses to this survey represents only a portion of the SBML-compatible software tools in the world, this is an exciting result. If the same proportions hold for the entire landscape of SBML-compatible software systems, then for computational scientists in biology today, the number of options available is surely a boon to research.

Simulation and analysis software are the most prominent in the results (see Fig. 1(b)). This is unsurprising considering that these areas historically have been the reasons for SBML's existence. On the other side of the spectrum, we are surprised by the relatively low number of tools that are categorized as annotation software. Although being able to create, edit and analyze a model are central goals in computational modeling, annotations are crucial to making a model interpretable and reusable by others (whether they are other humans, or other software) [24], [25]. We hope that support for working with annotations will increase in the future.

B. Mathematical frameworks in use

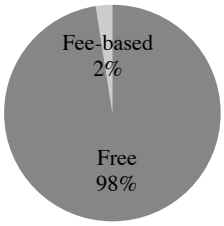
The results shown in Fig. 2 are a blend of unsurprising and surprising. It is unsurprising that the ordinary differential equation (ODE) framework is the most common among SBML-compatible software tools, simply because it is the most commonly-used framework in computational systems biology as a whole. It is also good to see a reasonable proportion of tools (28 out of 81, or $1/3$) support discrete stochastic simulation, an important framework for many research problems.

Some of the other frameworks concern areas that have not traditionally been SBML's forte, notably PDE and logical/Boolean networks. In fact, these are ongoing development areas for SBML today. The low showing for these frameworks may reflect a simple feedback loop: without adequate support in SBML, modelers may not have been easily able to express these models, which in turn led to software developers not being motivated to support these frameworks. With the introduction of SBML Level 3 Packages (such as the Spatial Package [26]), SBML will become better able to express these kinds of models, which will hopefully encourage the development of software support for the other frameworks.

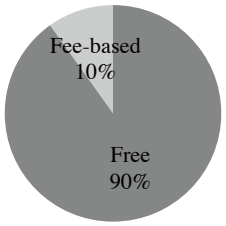
The surprisingly high number of "None of the above" responses prompted us to look at the answers written in the text field provided for elaborations. The most common answers given were: (i) steady-state or flux-balance analysis, and (ii) "not applicable" (most often given for format conversion tools). The former suggests that we need to add this as an option to a revised version of this survey.

C. SBML-specific software capabilities

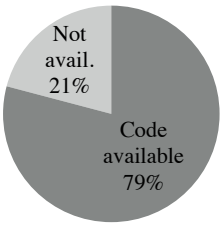
Turning now to the SBML-specific capabilities reported in Section III-C, we were surprised that a significant number of



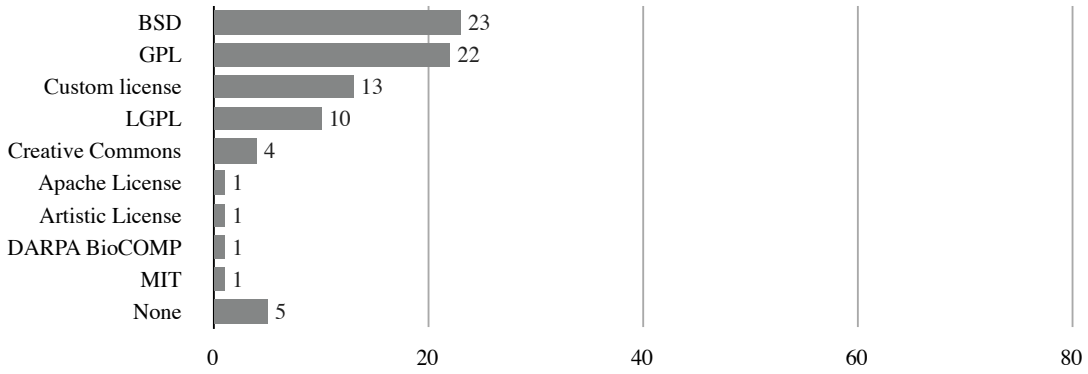
(a) Availability to academic users.



(b) Availability to non-academic users.



(c) Availability of source code.



(d) Distribution of software licenses used among the 81 software tools examined in this survey.

Fig. 8. Software availability and licensing.

tools understand SBML only in one direction, either reading *or* writing SBML, and not both. We examined the specific cases where tools only offered one direction, and found that most of them were conversion tools, model-generation tools, database systems, or simulation engines—in other words, either the starting point or the end point of common modeling workflows. (For example, databases are often the starting point of a workflow that moves models from retrieval, to simulation, to visualization, while simulation engines are often the end point.)

Support for SBML Level 3 stands at slightly over one third of the tools, as shown in Fig. 4. The Level 3 specification has been available for nine months (a draft version was available for a year longer). Software libraries compatible with Level 3 have existed since last year, and an updated libSBML [10] released at the same time as the specification was designed to make it easy to transition from SBML Level 2 to Level 3. We therefore hoped to see greater support for Level 3. On the other hand, we need to keep in mind two factors that may have influenced the lower than expected level of Level 3 adoption to date. First, many software tools are developed with limited funding and personnel. The degree of support for Level 3 today may thus not reflect a lack of desire on the part of developers, but simply a lack of resources during the past nine months. Second, there may also be a chicken-and-egg problem at work: tool developers may not be motivated to implement support for SBML Level 3 because there are not many Level 3 models "in the wild"—-but there may not be many Level 3 models available precisely because tool support is limited.

Finally, the pattern of results for the question about quantitative/mathematical understanding (see Fig. 5) left us puzzled. We could not understand how so few tools could *not* interpret basic SBML elements such as species and still consider themselves to be SBML-compatible tools. If we subtract the number of responses for "*Other, or not applicable*" (14) from the maximum (81), we are still left with 9 tools that did not indicate they could interpret the most basic SBML constructs—possibly indicating an inconsistency in the survey responses. We went back to the original question and data, and upon closer examination of the results, we noticed a pattern. The choices in the survey question were organized as follows:

1) *Interpret species, reactions, parameters, and/or compartments*
2) *Work with stoichiometric relationships/maps*
3) *Work with reaction kinetics*
4) *Work with other mathematical relationships*
5) *Work with conditional discontinuous events*
6) *Work with time delays*
7) *Other, or not applicable*

This sequence of questions had a relatively natural ordering in terms of complexity of mathematical understanding. When we examined the individual responses, we found that some respondents selected *only one* of the options, yet we knew from direct experience that the tools in question could *also* interpret the "simpler" SBML elements listed prior to their selections in the list of choices. In other words, it appeared some respondents misunderstood the instructions and did not check all the applicable choices, but rather *selected only the most complex option* that their software could work with.

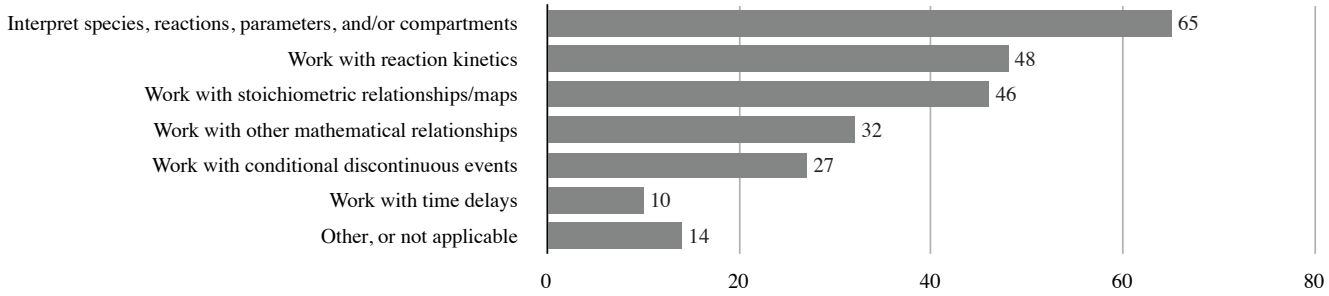


Fig. 9. Responses to the question from Fig. 5 after correcting for a likely misunderstanding in the instructions.

To correct for this, we generated a new table and graph by applying the following simple rule: if an entry had only one option selected from the list of choices above, we also checked all the prior options. For example, if a tool listed only "*Work with reaction kinetics*", we also checked "*Interpret species, reactions, parameters, and/or compartments*" and "*Work with stoichiometric relationships/maps*", because those appeared prior to "*Work with reaction kinetics*" in the list above.

Fig. 9 displays a graph of the results. Now the sum of responses for "*Interpret species, reactions, parameters, and/or compartments*" and "*Other, or not applicable*" add up to 79, which is more plausible. Examining the remaining two tools, we find that they selected multiple other capabilities such as (e.g., "*Work with other mathematical relationships*" and "*Work with conditional discontinuous events*") but not others. We plan on examining these tools more carefully to understand the operations they are performing and whether there remains a misunderstanding about this question in the survey.

D. Other standards supported

The results of Fig. 6 illustrates that SBML-compatible software tools support a number of other standards—clearly a benefit to software interoperability in general.

We are surprised at the low degree of support for MIRIAM [13], a basic standard for encoding annotations such as authorship information. The 16 responses (Fig. 6) represent less than one-fifth of the tools in this survey. We then realized that many tools are concerned with analysis and utility functions, and may not have any reason to operate on MIRIAM annotations. Limiting consideration to tools that specifically offer model creation capabilities (of which there are 50, according to Fig. 1(a)), we see the number for MIRIAM now represents nearly one third of the total—an improvement. We then examined specific cases of non-supporting tools, and found many that we suspect *could* support MIRIAM in at least a minimal fashion. It is possible that in some cases, the developers simply lack the resources to develop MIRIAM support, but in other cases, we suspect we need to do further outreach to promote the benefits of MIRIAM annotations.

E. General properties of the software implementations

Windows is the most widely-supported operating system (Fig. 7(a)), which is not surprising given that Windows has the dominant market share in operating systems. The results show, however, that Windows developers are not limiting their efforts to supporting *only* Windows, because the numbers indicate that few tools (a total of eight) are Windows-only.

What is perhaps more surprising is that Mac OS is in second place, edging out Linux. Many of the SBML-compatible software systems are developed in academic environments for academic users, where Linux is quite popular. We would have expected Linux to be the second-most popular choice. The result is even more surprising when one considers that none of the tools were designed *only* for the Mac OS environment.

Another pleasant surprise is that over half of the tools offer a means for interacting with the software through an application programming interface (API). This suggests that many developers are motivated to make their tools programmatically accessible, despite the extra effort normally required to implement this capability. This is gratifying to see, because both users and developers benefit from being able to join multiple tools together in pursuit of a research goal.

F. Availability and licensing

The survey reveals that software in this area is remarkably open and free. Fig. 8(a) shows that the overwhelming majority of the software is free to academic users, and Fig. 8(b) shows 90% of the software is likewise free to commercial users too.

The source code to most of the applications is available under open-source terms. The BSD [27] and GPL [28] licenses are the most popular, together accounting for over half of the software in the survey. Custom licenses are used for those tools that have restricted distribution terms. LGPL [29] and a smattering of other licenses account for most of the remainder.

We find the popularity of BSD to be interesting because it is more open and less restrictive than GPL. More generally, the fact that the majority of tools are available under any open-source terms at all suggests that developers genuinely want their software to be used *and reused* by others. Perhaps they recognize the positive feedback loop described in the introduction: better and more sophisticated tools enable better and deeper research, which in turn drives the evolution of new and more powerful software.

V. Conclusions

The survey results point to an abundance of software tools available today for computational systems biologists. The tools provide many capabilities, and nearly all are available under open-source terms. The simple fact that so many tools support SBML is a testament to SBML's utility and interoperability.

We would like to see greater software support for other related standards, especially MIRIAM and annotations. We believe that with greater support for these features will come even more powerful software systems in the future. Conversely, this survey revealed that many systems offer support for classes of modeling frameworks outside of SBML's core competence, which only serves to renew our sense of urgency to develop SBML Level 3 packages covering these other important areas.

Acknowledgments

We thank the SBML community, the survey respondents, the reviewers (for their comments), and NIH NIGMS (for support through grant R01 GM070923).

References

[1] M. Hucka, A. Finney, H. Sauro, H. Bolouri, J. Doyle, H. Kitano, A. Arkin, B. Bornstein, D. Bray, A. Cornish-Bowden, A. Cuellar, S. Dronov, E. Gilles, M. Ginkel, V. Gor, I. Goryanin, W. Hedley, T. Hodgman, J. Hofmeyr, P. Hunter, N. Juty, J. Kasberger, A. Kremling, U. Kummer, N. L. Novère, L. Loew, D. Lucio, P. Mendes, E. Minch, E. Mjolsness, Y. Nakayama, M. Nelson, P. Nielsen, T. Sakurada, J. Schaff, B. Shapiro, T. Shimizu, H. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang, "The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, pp. 524–31, 2003.

[2] M. Hucka, A. Finney, B. J. Bornstein, S. M. Keating, B. E. Shapiro, J. Matthews, B. L. Kovitz, M. J. Schilstra, A. Funahashi, J. C. Doyle, and H. Kitano, "Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project," *IEE Systems Biology*, vol. 1, no. 1, pp. 41–53, 2004.

[3] SBML Team, "SBML.org," Available at http://sbml.org, 2011.

[4] ——, "The SBML software guide," Available at http://sbml.org/SBML_Software_Guide, 2011.

[5] 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. Bergmann, R. Gauges, P. Ghazal, H. Kawaji, L. Li, Y. Matsuoka, A. Villéger, S. E. Boyd, L. Calzone, M. Courtot, U. Dogrusoz, T. C. Freeman, A. Funahashi, S. Ghosh, A. Jouraku, S. Kim, F. Kolpakov, A. Luna, S. Sahle, E. Schmidt, S. Watterson, G. Wu, I. Goryanin, D. B. Kell, C. Sander, H. Sauro, J. L. Snoep, K. Kohn, and H. Kitano, "The Systems Biology Graphical Notation," *Nature Biotechnology*, vol. 27, no. 8, pp. 735–741, Aug. 2009.

[6] D. Köhn and N. Le Novère, "SED-ML – an XML format for the implementation of the MIASE guidelines," in *Computational Methods in Systems Biology*, ser. Lecture Notes in Computer Science, M. Heiner and A. Uhrmacher, Eds. Springer, 2008, vol. 5307, ch. 15, pp. 176–190.

[7] SBML Team, "SBML software details questionnaire," http://sbml.org/SBML_Software_Guide/SBML_Software_Details_Questionnaire, 2011.

[8] SurveyMonkey, "SurveyMonkey," http://www.surveymonkey.com, 2011.

[9] M. Hucka, F. T. Bergmann, S. Hoops, S. M. Keating, S. Sahle, J. C. Schaff, L. P. Smith, and D. J. Wilkinson, "The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core," Available at http://sbml.org/Documents/Specifications, 2010.

[10] B. J. Bornstein, S. M. Keating, A. Jouraku, and M. Hucka, "LibSBML: an API library for SBML," *Bioinformatics*, vol. 24, pp. 880–881, 2008.

[11] M. J. Herrgård, N. Swainston, P. Dobson, W. B. Dunn, K. Y. Arga, M. Arvas, N. Blüthgen, S. Border, R. Costenoble, M. Heinemann, M. Hucka, N. Le Novère, P. Li, W. Liebermeister, M. L. Mo, A. P. Oliveria, D. Petranovic, S. Pettifer, E. Simeonidis, K. Smallbone, I. Spasić, D. Weichart, R. Brent, D. S. Broomhead, H. V. Westerhoff, B. Krdar, M. Penttilä, E. Klipp, B. Ø. Palsson, U. Sauer, S. G. Oliver, P. Mendes, J. Nielsen, and D. B. Kell, "A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology," *Nature Biotechnology*, vol. 26, no. 10, pp. 1155–1160, 2008.

[12] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. S. Marshall, "Graphml progress report: Structural layer proposal," in *Proc. of the 9th International Symposium on Graph Drawing*. Springer-Verlag, 2002.

[13] N. L. Novère, A. Finney, M. Hucka, U. S. Bhalla, F. Campagne, J. Collado-Vides, E. J. Crampin, M. Halstead, E. Klipp, P. Mendes, P. Nielsen, H. Sauro, B. Shapiro, J. L. Snoep, H. D. Spence, and B. L. Wanner, "Minimum information requested in the annotation of biochemical models (MIRIAM)," *Nature Biotechnology*, vol. 23, no. 12, pp. 1509–1515, Dec. 2005.

[14] N. Le Novère, M. Courtot, and C. Laibe, "Adding semantics in kinetics models of biochemical pathways," in *2nd Int'l. ESCEC Workshop on Experimental Standard Conditions on Enzyme Characterizations*, C. Kettner and M. G. Hicks, Eds. Beilstein Institut, 2006, pp. 137–153.

[15] E. Demir, M. P. Cary, S. Paley, K. Fukuda, C. Lemer, I. Vastrik, G. Wu, P. D'Eustachio, C. Schaefer, J. Luciano, F. Schacherer, I. Martinez-Flores, Z. Hu, V. Jimenez-Jacinto, G. Joshi-Tope, K. Kandasamy, A. C. Lopez-Fuentes, H. Mi, E. Pichler, I. Rodchenkov, A. Splendiani, S. Tkachev, J. Zucker, G. Gopinath, H. Rajasimha, R. Ramakrishnan, I. Shah, M. Syed, N. Anwar, O. Babur, M. Blinov, E. Brauner, D. Corwin, S. Donaldson, F. Gibbons, R. Goldberg, P. Hornbeck, A. Luna, P. Murray-Rust, E. Neumann, O. Reubenacker, M. Samwald, M. van Iersel, S. Wimalaratne, K. Allen, B. Braun, M. Whirl-Carrillo, K.-H. H. Cheung, K. Dahlquist, A. Finney, M. Gillespie, E. Glass, L. Gong, R. Haw, M. Honig, O. Hubaut, D. Kane, S. Krupa, M. Kutmon, J. Leonard, D. Marks, D. Merberg, V. Petri, A. Pico, D. Ravenscroft, L. Ren, N. Shah, M. Sunshine, R. Tang, R. Whaley, S. Letovksy, K. H. Buetow, A. Rzhetsky, V. Schachter, B. S. Sobral, U. Dogrusoz, S. McWeeney, M. Aladjem, E. Birney, J. Collado-Vides, S. Goto, M. Hucka, N. Le Novère, N. Maltsev, A. Pandey, P. Thomas, E. Wingender, P. D. Karp, C. Sander, and G. D. Bader, "The BioPAX community standard for pathway data sharing." *Nature biotechnology*, vol. 28, no. 9, pp. 935–942, Sep. 2010.

[16] C. M. Lloyd, M. D. Halstead, and P. F. Nielsen, "CellML: its future, present and past," *Prog. Biophysics & Molecular Biology*, vol. 85, 2004.

[17] M. Kanehisa and S. Goto, "KEGG Markup Language," Available at http://www.genome.jp/kegg/xml/, 2010.

[18] P. Gleeson, S. Crook, R. C. Cannon, M. L. Hines, G. O. Billings, M. Farinella, T. M. Morse, A. P. Davison, S. Ray, U. S. Bhalla, S. R. Barnes, Y. D. Dimitrova, and R. A. Silver, "NeuroML: A language for describing data driven models of neurons and networks with a high degree of biological detail," *PLoS Computational Biology*, vol. 6, no. 6, p. e1000815, 2010.

[19] H. Yun, D.-Y. Lee, J. Jeong, S. Lee, and S. Y. Lee, "MFAML: a standard data structure for representing and exchanging metabolic flux models," *Bioinformatics*, vol. 21, no. 15, pp. 3329–3330, 2005.

[20] J. Billington, S. Christensen, K. van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. Weber, "The Petri Net Markup Language: Concepts, Technology, and Tools," in *Applications and Theory of Petri Nets 2003: 24th Int. Conf.*, Jun. 2003, pp. 1023–1024.

[21] M. Galdzicki, C. Rodriguez, D. Chandran, H. M. Sauro, and J. H. Gennari, "Standard Biological Parts Knowledgebase," *PLoS ONE*, vol. 6, no. 2, pp. e17 005+, Feb. 2011.

[22] MathWorks, Inc., "MATLAB," Available at http://mathworks.com, 2011.

[23] R Development Core Team, *R: A Language and Environment for Statistical Computing*, Available at http://www.R-project.org, 2011.

[24] C. Li, M. Donizelli, N. Rodriguez, H. Dharuri, L. Endler, V. Chelliah, L. Li, E. U. He, A. Henry, M. I. Stefan, J. L. Snoep, M. Hucka, N. Le Novère, and C. Laibe, "Biomodels database: An enhanced, curated and annotated resource for published quantitative kinetic models," *BMC Systems Biology*, vol. 4, 6 2010.

[25] J. Saez-Rodriguez, L. G. Alexopoulos, and G. Stolovitzky, "Setting the standards for signal transduction research," *Science Signaling*, vol. 4, no. 160, p. pe10, 2011.

[26] SBML Team, "SBML Level 3 Status Summary Table," Available at http://sbml.org/Community/Wiki, 2011.

[27] Linux Information Project, "BSD license definition," Available at http://www.linfo.org/bsdlicense.html, 2011.

[28] Free Software Foundation, "GNU General Public License," Available at http://www.gnu.org/licenses, 2011.

[29] ——, "GNU Lesser General Public License," Available at http://www.gnu.org/licenses/, 2011.