The JigCell Model Builder: A Spreadsheet Interface for Creating Biochemical Reaction Network Models

Marc T. Vass, Clifford A. Shaffer, Naren Ramakrishnan, Layne T. Watson, and John J. Tyson

Abstract—Converting a biochemical reaction network to a set of kinetic rate equations is tedious and error prone. We describe known interface paradigms for inputing models of intracellular regulatory networks: graphical layout (diagrams), wizards, scripting languages, and direct entry of chemical equations. We present the JigCell Model Builder, which allows users to define models as a set of reaction equations using a spreadsheet (an example of direct entry of equations) and outputs model definitions in the Systems Biology Markup Language, Level 2. We present the results of two usability studies. The spreadsheet paradigm demonstrated its effectiveness in reducing the number of errors made by modelers when compared to hand conversion of a wiring diagram to differential equations. A comparison of representatives of the four interface paradigms for a simple model of the cell cycle was conducted which measured time, mouse clicks, and keystrokes to enter the model, and the number of screens needed to view the contents of the model. All four paradigms had similar data entry times. The spreadsheet and scripting language approaches require significantly fewer screens to view the models than do the wizard or graphical layout approaches.

Index Terms—Biochemical reaction networks, bioinformatics, modeling, user interface paradigms.

1 Introduction

Regulatory network models attempt to deduce the physiological properties of a cell from wiring diagrams of its control systems. These networks of interacting proteins are intrinsically dynamic. They describe the molecular mechanisms by which a cell changes in space and time to respond to stimuli, grow and reproduce, differentiate, and do all the other remarkable tricks that are necessary to stay alive and propagate the species.

A simple example of a regulatory network is the set of reactions controlling the activity of MPF (mitosis promoting factor) in Xenopus oocyte extracts [13], which we refer to herein as the frog egg model (see Fig. 1). Such networks are often represented as graphs where vertices represent substrates and products (collectively referred to as species) and labeled directed edges connecting vertices represent the reactions. Chemical reactions cause the concentrations of the chemical species (C_i) to change in time according to the equation

$$\frac{dC_i}{dt} = \sum_{j=1}^{R} b_{ij} v_j, i = 1, \ldots, N,$$

where R is the number of reactions, v_j is the velocity of the jth reaction in the network, and b_{ij} is the stoichiometric coefficient of species i in reaction j ($b_{ij} < 0$ for substrates, $b_{ij} > 0$ for products, $b_{ij} = 0$ if species i does not take part in reaction j).

The full set of rate equations is a mathematical representation of the temporal behavior of the regulatory network. Modelers are faced with many computational problems: accurately and efficiently solving equations when velocities are characterized by widely varied time constants, finding steady state solutions, estimating rate constants by fitting numerical solutions to experimental data, and identifying bifurcation points in the multidimensional parameter space.

For example, a realistic model of the budding yeast cell cycle consists of about 30 differential equations containing 100 rate constants [5]. The parameters are estimated from the cell-cycle behavior of more than 100 mutants defective in the regulatory network. Simulating the entire set takes from a few minutes to an hour on a desktop PC for one choice of kinetic constants. To fit the model to the mutant data by nonlinear regression will likely require thousands of repetitions of the full calculations. A model of such complexity (10-100 equations) represents the upper limit of what a dedicated modeler can produce "by hand" with a good numerical integrator like LSODE [17]. To adequately describe fundamental physiological processes (such as the control of cell division) in mammalian cells will require models of at least 100-1,000 equations. To handle this next generation of dynamical models will require sophisticated software to automate the modeling cycle: network specification, equation generation, simulation and data management, and parameter estimation. Ongoing efforts such as the DARPA BioSPICE initiative [6] and the DOE Genomes to Life project [7] aspire to support models at least one order of magnitude larger than are currently used.

- *M.T. Vass, C.A. Shaffer, N. Ramakrishnan, and L.T. Watson are with the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106. E-mail: {mvass, shaffer, naren, ltw}@vt.edu.*
- *J.J. Tyson is with the Department of Biology, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106. E-mail: tyson@vt.edu.*

Manuscript received 11 July 2005; accepted 3 Oct. 2005; published online 1 May 2006.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0078-0705.

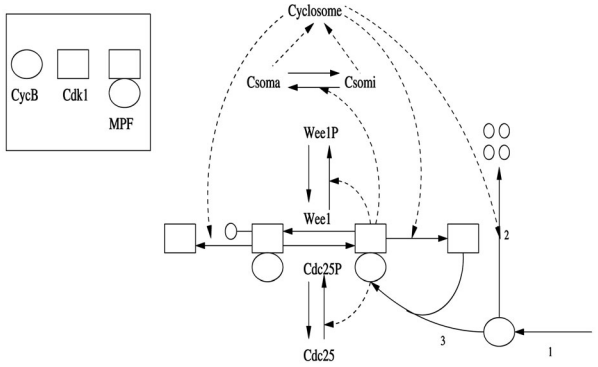


Fig. 1. Pathway diagram for the frog egg cell cycle. Cyclin B, synthesized in reaction 1, combines with Cdk1 (reaction 3) to form active MPF. MPF is inactivated by phosphorylation of the Cdk1 subunit by Wee1. Cdc25P reverses the phosphorylation step, converting inactive MPF back to active MPF. Finally, a protein complex (called the cyclosome) degrades cyclin B protein (reaction 2).

Until recently when the current generation of tools were developed, many stages in the modeling cycle had been done by hand. This cycle typically begins with the modeler drawing a wiring diagram (see Fig. 1), then deriving a set of corresponding reaction equations, and finally converting the reaction equations to differential equations (in a format appropriate to simulate the equations). This presents two problems. First, it consumes a great amount of time and effort on the part of the researchers to convert their intuitive concept of the model into a suitable set of reaction equations. Second, there are many opportunities for errors to creep in, especially at the (essentially mechanical) step of converting reaction equations to differential equations.

A wiring diagram, like Fig. 1, nicely represents the topology of a reaction network (reactants, products, enzymes). But, one cannot also specify the kinetics of the network (the reaction rate laws, v_j) adjacent to the reaction arrows without cluttering the diagram beyond recognition. A large reaction network can become so complex that even its topological features are obscured by a large number of intersecting arrows. Obscurity is increased further because there is no standard format for drawing such graphs, though Kohn's notation [11] has much to recommend it. Without precise notational conventions, it is impossible to convert a wiring diagram into an unambiguous model, either by hand or machine.

A second method for representing a reaction network is to explicitly write out the chemical reactions, $S_1, \ldots, S_i$ into products $P_1, \ldots, P_j$. This loses some of the intuitive appeal of the diagrammatic approach, but allows for a more compact definition of a reaction network. Normally, the modeler has already made a hand or CAD-drawn version of the network in graphical form, showing the interactions in a qualitative sense but without the quantitative information of the velocity equations or the parameter values.

Models often include concepts not captured by the differential equations alone. 1) Linear conservation relations are defined by linear combinations of species concentrations that remain constant throughout a simulation: $\sum_{i=1}^{N} a_i C_i(t) = \text{constant}$, $a_i = \text{constant}$. Such constraints arise from linear dependencies in the stoichiometry matrix: $\sum_{i=1}^{N} a_i b_{ij} = 0$. 2) Events are special actions that trigger in the model under given conditions. For example, cellular division could be represented by a decrease in mass and might occur when a given function involving some number of chemical species reaches a threshold during a simulation. Neither a network diagram nor a chemical reaction can represent such events.

The rest of this paper is organized as follows: Section 2 describes three user interface paradigms for creating computer models of biochemical reaction networks. Section 3 describes in detail the JigCell Model Builder, which uses a spreadsheet interface for network specification. Section 4 reports the results of two empirical usability studies of network model entry and inspection by different methods. The first compares the JigCell Model Builder spreadsheet interface to manual creation of the ordinary differential equation model (the approach used by our modelers prior to creating the JigCell Model Builder). The second compares all four interface paradigms in terms of quantitative measures for work that are not dependent on subject experience or capability. Some conclusions regarding computer support environments for regulatory network modeling are offered in Section 5.

2 User Interface Paradigms for Model Builders

In this section, we describe four approaches to designing a user interface for specifying regulatory network models: a graphical wiring diagram, a "wizard," a script, and a spreadsheet. We illustrate each approach with existing systems, and compare them for their gross characteristics affecting usability and potential for user satisfaction.

2.1 Graphical Interfaces

Virtual Cell [24], [12] and JDesigner [21] are examples of systems whose user interface is based on specifying a model as a graphical network. In Virtual Cell, for example, the network is created in a workspace where a species is represented as a circle and a chemical reaction is represented by a barbell. Substrates are connected to the left end of the barbell and products to the right end. Catalysts are connected to the middle of the barbell. Right clicking the barbell generates a dialog box that allows a user to enter the velocity of the reaction. It is typical for graphical approaches to remove the rate law and constants from the graph for visual clarity, at the expense of requiring the user to access multiple screens to see and edit this information. The mass action rate law is used by Virtual Cell as a default, with a locally defined rate law being allowed but no global user-defined rate law may be specified.

With all existing diagram editors, as the model grows, an increasing fraction of the graph's edges cross one another, leading to confusion for the user. Ideally, model complexity would be dealt with by some bundling mechanism whereby sections of the graph can be replaced by a single icon, to be contracted or expanded as desired. While there is not yet consensus on how best to do this and much work remains on developing such aggregation mechanisms, several systems now support some form of aggregation [4], [8], [25].

Users must typically resort to some alternative interface for entering the inherently textual information required to

specify the reaction equations, such as a popup text entry box or alternate screens. No existing graphical interfaces are purely graphical for this reason. Entering the various parameter values and rate equations can account for a significant fraction of the total model description. However, the graphical approach provides modelers with the intuitive appeal of being closely in line with their mental model of the system.

2.2 Wizard Interfaces

The "wizard" metaphor consists of a series of dialog boxes that lead a user through the various stages of creating a reaction network. Wizard interfaces are a well-known approach for guiding users through a highly structured and well-defined task, such as entering information required to prepare income tax forms, and for installing software or hardware in Microsoft Windows. Gepasi [14], [15] exemplifies the wizard interface for regulatory modeling. First, the user enters the chemical equations for a network by selecting the *reactions* button. Next, to specify the velocities of the equations, the user clicks the *kinetics* button and selects the rate law to use for the chemical equation. To add a user-defined rate law, the user must go back to the previous screen and click the *kinetic types* button and choose the *add* button. User-defined functions are specified in a similar manner by clicking on the *functions* button and choosing the *add* button.

The advantage of the wizard interface is that it guides the user through the model-building process. This provides more support for users unfamiliar with either the tool or the modeling process. However, a wizard interface breaks the model representation (and the model building process) across many screens. As a consequence, models are difficult to visualize as a whole when using the wizard paradigm. Hence, the modeler might lose focus on the full model and be distracted by navigating through a series of dialogs showing only detailed portions of the model at any instant. The wizard approach is likely to lead to more mouse clicks and movement than would be necessary in an interface more tuned to an experienced user, so it is inherently inefficient to some degree. However, if the data entry task can be sufficiently well structured, then the user can be kept on target entering the information most immediately needed. An alternative display technique could be used to view a model once it has been entered, just as an income tax program will display the filled-out form once data entry has been completed.

2.3 Script-Based Interfaces

A third approach for building reaction networks is direct editing of the collection of reaction equations, expressed in ASCII text. Examples of this approach are SCAMP [20] and Jarnac [19].

Jarnac uses a compact text-based scripting language similar to the specification of chemical equations. A large model can be specified quickly and can be compartmentalized. Compartmentalized models become models unto themselves, able to be viewed as a black box with inputs and outputs. This removal of data overload allows the user to more easily visualize large models. Because Jarnac uses a programming language-like form, it is compact and precise. However, the concept of "programming" the model could be unnatural for many life scientists.

No system is entirely restricted to one paradigm. For example, Virtual Cell uses a scripting language to make up for limitations to its graphical interface. Some systems attempt to combine the strengths of the graphical approach with the equation editing approach. Jarnac works in combination with JDesigner (JDesigner is intended as a graphical front end to Jarnac).

With the JigCell Model Builder (JCMB), we introduce an important interface variation based on inputting models using direct entry of chemical equations. JCMB specifies reaction networks using a spreadsheet paradigm. Biologists are more likely to be experienced already with creating spreadsheets than with creating programs or scripts. The spreadsheet approach also has the advantage that it imposes more structure onto the equation-writing process than a scripting language. JCMB is part of the JigCell system for modeling and analyzing biochemical reaction networks [1], [2], [26]. JigCell also uses spreadsheet-based interfaces for describing ensembles of simulation runs and comparing experimental data to simulation results.

3 The JigCell Model Builder

JCMB (see Fig. 2) mimics the standard functionality of a spreadsheet, with each reaction, function, and rate law being defined on a separate row. A design goal of JCMB is to reduce the number of errors generated in the modeling process. The spreadsheet interface allows the modeler to visualize the entirety of many current models and allows expression of models in the language of the domain (as reaction equations). Modelers can see and specify a chemical equation and its associated rate law and constants on the same line.

Typically, the user types in the reaction equations while frequently consulting a wiring diagram, such as Fig. 1. This process is error prone for many reasons. Typos and copy-paste mistakes are inevitable because of the tedium and repetitiveness of the process. Other errors arise from misspecifying the rate law of a reaction, neglecting entire terms from the right-hand side of the fundamental equation, and from mistaken implementation of conservation conditions. If a conservation condition is overlooked, then the resulting set of differential equations contains redundancies and round-off errors in simulations can lead to significant violations of the law of conservation of mass. More seriously, human beings may mistakenly identify or apply conservation conditions, leading to differential equations that are incorrect.

JCMB attempts to reduce these errors by forcing users to adopt a reaction-centered approach that separates a reaction from its rate law specification. This allows the computer to apply the specified rate law to discover the velocity for a particular reaction, which is then shown to the user in a separate column.

Errors in a model lead to meaningless simulation results. Such bugs are difficult to detect and might cost the modeler hours searching for errors caused by syntactic slips made at the model-building stage. JCMB attempts to disallow inconsistencies in a model while it is being entered. For

#	Reaction	Name	Type	Equation	Parameters
1	->CycB		Mass Action	k1	Kf=k1
2	CycB->		Mass Action	Cyclosome*CycB	Kf=Cyclosome
3	CycB+Cdk1->MPFa		Mass Action	k3*CycB*Cdk1	Kf=k3
4	MPFa->MPFi		Mass Action	V(kwp,Wee1p,kwpp,Wee1)*MPFa	Kf=V(kwp,Wee1p,kwpp,Wee1)
5	MPFi->MPFa		Mass Action	V(kcp,Cdc25p,kcpp,Cdc25)*MPFi	Kf=V(kcp,Cdc25p,kcpp,Cdc25)
6	Cdc25p->Cdc25		Michaelis-Menten	(k25r*Cdc25p*1)/(J25r+Cdc25p)	M1=1; J1=J25r; k1=k25r
7	Cdc25->Cdc25p		Michaelis-Menten	(k25f*Cdc25*(MPFa+eps1))/(J25f+Cdc25)	M1=MPFa+eps1; J1=J25f; k1=k25f
8	Wee1->Wee1p		Michaelis-Menten	(kweef*Wee1*(MPFa+eps2))/(Jweef+Wee1)	M1=MPFa+eps2; J1=Jweef; k1=kweef
9	Wee1p->Wee1		Michaelis-Menten	(kweer*Wee1p*1)/(Jweer+Wee1p)	M1=1; J1=Jweer; k1=kweer
10	Csoma->Csomi		Michaelis-Menten	(kcyr*Csoma*1)/(Jcyr+Csoma)	M1=1; J1=Jcyr; k1=kcyr
11	Csomi->Csoma		Michaelis-Menten	(kcyf*Csomi*(MPFa+eps3))/(Jcyf+Csomi)	M1=MPFa+eps3; J1=Jcyf; k1=kcyf
12	MPFa->Cdk1		Mass Action	Cyclosome*MPFa	Kf=Cyclosome
13	MPFi->Cdk1		Mass Action	Cyclosome*MPFi	Kf=Cyclosome
14		V	Function	A1*A2+A3*A4	
15	Cyclosome		Species	V(k2p,Csomi,k2pp,Csoma)	k2pp=k2pp; k2p=k2p

Fig. 2. Frog egg extract model in JCMB.

example, the user does not type a rate law, but, instead, selects one from a pulldown list. As another example, users do not determine conservation relationships between species. Instead, JCMB does this automatically and all the user may do is reorder the relationships. When an inconsistency or error is detected (such as a syntactic error in a cell), the spreadsheet highlights the problem cell in orange and propagates the error throughout the spreadsheet by highlighting dependent problem cells as well. These errors are detected by the program checking each cell for mathematical validity within its context whenever it has been changed. Once the error has been fixed, all cells now made consistent will return to their normal color.

JCMB also attempts to reduce errors and structure the editing process by restricting the ability to edit columns where appropriate. For example, since the *equation* column of a reaction is derived from its type and parameters, this column may not be edited for *reaction* rows (in contrast, it may be edited for a *function* row). The *name* column of various row types is edited when it is first created, but may not be edited afterward.

The spreadsheet approach appears to pack more information on the screen than does the graphical approach, thus allowing the user to have access to a greater portion of the model at one time (or the entire model for smaller examples). Of course, there is a limit to how far this can go and larger models will require the user to scroll over multiple screens of information. It was observed earlier that, as a model grows, the graphical approach will require more edges that cross each other. For the spreadsheet, an equivalent breakdown is more reference to variables not on the current screen. As with the graphical approach, larger models presented in a spreadsheet view would benefit from some sort of aggregation or component mechanism. This might express itself as a reference to components similar to subroutine calls in a programming language.

With respect to structuring of the data entry task, the spreadsheet appears more structured than the graphical and scripting approaches and less structured than the wizard. The spreadsheet (being based on reaction equations) appears to be further from the user's mental model than the wiring diagram of the graphical approach, but closer than the abstract data collection mechanism of the wizard.

One inherent shortcoming of the spreadsheet approach is that columns and their names affect all row types. For example, the "Name" column means different things, depending on the row type, and this column title is vague because of its multiple uses. The column order is also fixed for all rows, although JCMB permits the user to reorder the columns (for the entire spreadsheet) at will.

We briefly describe the major row types that might appear in the main JCMB reaction spreadsheet. We then describe the auxiliary spreadsheets that JCMB uses to organize data other than reactions.

3.1 Reaction Row

A reaction row specifies what species are involved in a chemical reaction, the chemical equation for the reaction, the rate law, and the modifiers and constants needed by the rate law. A reaction equation consists of a list of substrates separated by $+$s, an arrow "$\rightarrow$," and a list of products, also separated by $+$s. The stoichiometry of a species is specified by placing the value directly in front of the name, or by writing the value separated from the name with a "*."

The Type column specifies the rate law to be applied for the reaction given in the reaction column for this row. The three predefined rate laws are Mass action, Michaelis-Menten, and local (we describe below how to define new rate laws). Mass action is defined as $v = k_f \Pi_i S_i^{b_i}$, where the arguments to the rate law are the concentrations of the substrates (S_i), the stoichiometric coefficients (b_i), and a constant k_f. For reaction $A \rightarrow B$ with concentrations $[A]$ and $[B]$, respectively, carried out by enzyme M_1, the Michaelis-Menten rate law is defined as

$$\frac{d[B]}{dt} = (K_1 * M_1 * [A])/(J_1 + [A]).$$

The term "local" in the *Type* column indicates a function not reused elsewhere (see Fig. 8). This may contain any algebraic expression that uses species given in the *Reaction*

#	Reaction	Name	Type	Equation	Parameters
1		degradation	New	S1*k1*(E1+k2*E2)	
2		catalyzed	New	S1*k*E1	

Fig. 3. Rate law rows.

column, any constant/modifier defined in the *Parameters* column, any function defined by the user in the current model, or any predefined function. Any symbol that is not defined in the model will be regarded as a constant/modifier and will appear in the *Parameters* column for this reaction. Local functions are useful if the equation is likely not to be used again within a model, as it avoids the definition of a new rate law for a single reaction.

The *Equation* column is not editable by the user unless the row has a local rate law. Otherwise, the column will display the expression derived from substituting the values for the constants, modifiers, and species into the rate law given in the *Type* column.

The *Parameters* column lists the modifiers and constants that exist for the given rate law. Where the user has specified the values of the modifier or constant to be an argument to the rate law, it is shown on the right hand side of the "=" sign next to the name of the rate law argument. To specify the values for the rate law arguments, the user can click on this column, which will display a window for editing the values.

3.2 Rate Law Row

A rate law (Fig. 3) specifies the velocity of a chemical reaction by providing a unique name and the associated equation for the rate of the reaction. It is created by selecting *new* as the row type. Once a new rate law is defined, it will become available within the current model for use in reaction rows. The *Reaction* column has no meaning in a rate law, so it must remain empty. The *Equation* column specifies the algebraic equation defining the rate law. Substrates and Products for the rate law are specified as Si or Pi, respectively, where i is the order in which the species appears in the list of substrates and products. User-defined functions and predefined functions may be used in this column. Any variable other than a Si or Pi will be shown in the *Parameters* column for any reaction using this rate law. The *Parameters* column is not editable by the user for rate law rows. Once a rate law has been defined, the *name* column is no longer editable. Line 14 of Fig. 2 illustrates a function definition, which is used in lines 4, 5, and 15.

3.3 Function Row

A function row specifies an algebraic function that takes a list of arguments and returns a value. The *reaction* column has no meaning in a function (Fig. 4), so it must remain empty. The *name* column specifies the function name. A function name may not be duplicated by other functions, species equations, or species in a reaction. The user must choose *Function* in the *Type* column.

#	Reaction	Name	Type	Equation	Parameters
1		BB	Function	A2-A1+A3*A2+A4*A1	

Fig. 4. Function row.

#	Name	Initial Value
1	bck0	0.054
2	bub2h	1.0
3	bub2l	0.2
4	CDC15T	1.0
5	CDH1T	1.0
6	CLN3MAX	0.4
7	Dn3	1.0
8	ebudb5	1.0
9	ebudn2	0.16
10	ebudn3	0.05
11	ec1b2	0.4
12	ec1b5	0.25
13	ec1k2	0.03
14	ec1n2	0.038
15	ec1n3	0.3
16	ef6b2	0.35
17	ef6b5	0.13
18	ef6k2	0.03

Fig. 5. Constants.

The *Equation* column specifies the algebraic equation $y = f(A_1, \ldots, A_n)$ defining the function whose return value is y. Arguments for the function are named as Ai, where i is the order in which the argument appears in the list of arguments to the function. User-defined functions and predefined functions may be used in this column. Any variable not of the form Ai is assumed to be a constant or modifier and will be shown in the *Parameters* column for this function. This column lists the modifiers and constants that exist for the given equation and follows the same rules as in the *Reaction* row.

3.4 Species Equation Row

A species equation row (see, for example, line 15 in Fig. 2) specifies the equation for a chemical species that does not appear as a substrate or product in any reaction. In this example, the species row expresses the amount of cyclosone present in the model. The name provided in the reaction column is the name of the species; once set, it cannot be changed because other rows depend on it. This name may also represent an intermediate variable for use in computation. The equation column is defined by the user, similar to a *Reaction* row. The *Parameters* column lists the modifiers and constants specified by the equation and follows the same rules as in the *Reaction* row.

While the bulk of the user's attention will be on the main reaction spreadsheet, JCMB includes four more spreadsheets to organize information separate from reactions. Note that the process of editing the reactions spreadsheet will in turn update these auxiliary spreadsheets. For example, adding new constants or species to the reaction spreadsheet will cause new entries to be added to the constant or species spreadsheet, respectively.

3.5 Constants and Species Spreadsheets

The constants spreadsheet (Fig. 5) contains all symbols that have not been recognized as a species and whose values must be specified. The list is generated automatically from the *Model* spreadsheet. The species spreadsheet is similar to the constants spreadsheet and contains all species from the reaction rows specified in the Model spreadsheet. It allows the user to specify the initial conditions for the species.

#	Conservation Relation	Constant Total Name	Dependent Species
1	Ci + Ca	CTotal	Ca
2	Wi + Wa	WTotal	Wa
3	Mi + Ma	TotalCyclin	Ma

Fig. 6. Conservation relations.

3.6 Conservation Relation Spreadsheet

Conservation relations (Fig. 6) appear in their own spreadsheet. The *Conservation Relation* column shows the various conservation relations that exist in the model and is filled in automatically by the Model Builder using Reder's method [18], [23], [22]. The *Constant Total Name* column is editable by the user and is for giving a name to the constant total for the conservation relation. This name will appear as a constant in the *Constants* spreadsheet.

The *Dependent Species* column is editable by the user and must contain the name of a species from this relation that is to be treated as dependent. This means that JCMB will not generate a differential equation for this species and will, instead, use a linear combination of other species to generate its concentration. JCMB automatically chooses one of the species to be dependent by default.

3.7 Events Spreadsheet

Events (Fig. 7) are conditions that, when met, trigger certain user-defined actions. The *Name* column is a name for the given event. The *Trigger* column may contain any combination of algebraic expressions and Boolean operators that evaluate to a Boolean value. If the expression evaluates to true from being previously false, the assignments are performed and the event has occurred. The *Delay* column specifies a delay, which is the amount of simulation time to wait after the event has occurred before the changes listed in the *Assignments* column are applied. The *Assignments* column specifies a list of species and constants that are to be changed, along with their new values, when the desired condition has been met.

#	Name	Trigger	Delay	Assignments
1		(SPN-1.0)>0.0	0.0	MAD2=Mad2I; Ite1=Ite1h; bub2=bub2I

Fig. 7. Event spreadsheet.

4 Evaluation

We performed two sets of empirical studies. We first compared JCMB against hand conversion of a set of reactions to ODEs. We then compared the four interface paradigms for their ability to support model entry and model viewing.

4.1 Effectiveness of JCMB as an Error-Reducing Editor

JCMB has been evaluated [27] to determine its effectiveness in reducing errors in converting network diagrams to differential equations and to classify the types of errors made in the use of JCMB. While one might simply assume that a tool for this purpose will be better than hand conversion, this should by no means be taken for granted. Many experimental software tools do not, in fact, provide

#	Reaction	Name	Type	Equation	Parameters
1		degradation	New	S1*k1*(E1+k2*E2)	
2		catalyzed	New	S1*k*E1	
3		GK	Function	2*A1*A4/((A2-A1+A2*A3+A1*A4)+...	
4	->W		Local	k1	k1=k1synth
5	Y->W		Mass Action	kdiss*Y	Kf=kdiss
6	Z->W		Mass Action	kdiss*Z	Kf=kdiss
7	X->W		Mass Action	kdiss*X	Kf=kdiss
8	W->X		Mass Action	k1*W	Kf=k1
9	W->		degradation	W*k4*(Y+ro*TY)	E1=Y; k2=ro; E2=TY; k1=k4
10	Z->X		Mass Action	k6*Z	Kf=k6
11	Z->		degradation	Z*k4*(Y+ro*TY)	E1=Y; k2=ro; E2=TY; k1=k4
12	X->Z		catalyzed	X*k5*KIN	E1=KIN; k=k5
13	Z+XIC->TZ		Mass Action	kb*Z*XIC	Kf=kb
14	TZ->XIC+Z		Mass Action	ku*TZ	Kf=ku
15	TZ->Z		Mass Action	ktt*TZ	Kf=ktt
16	TZ->XIC		degradation	TZ*k4*(Y+ro*TY)	E1=Y; k2=ro; E2=TY; k1=k4
17	X+XIC->TX		Mass Action	kb*X*XIC	Kf=kb
18	TX->XIC+X		Mass Action	ku*TX	Kf=ku
19	TX->X		Mass Action	ktt*TX	Kf=ktt
20	TX->XIC		degradation	TX*k4*(Y+ro*TY)	E1=Y; k2=ro; E2=TY; k1=k4
21	Y+XIC->TY		Mass Action	kb*Y*XIC	Kf=kb
22	TY->XOC+Y		Mass Action	ku*TY	Kf=ku
23	TY->XIC		degradation	TY*k4*(Y+ro*TY)	E1=Y; k2=ro; E2=TY; k1=k4
24	TY->Y		Mass Action	ktt*TY	Kf=ktt
25	Y->		degradation	Y*k4*(Y+ro*TY)	E1=Y; k2=ro; E2=TY; k1=k4
26	Y->X		Local	Y*k*GK(k1,k2,J1,J2)	k=k3; J2=Jeinv; k2=keinv*(Y+ro*TY); J1=Je; k1=ke
27	X->Y		Local	X*(k1*(E1+k2*E2)+k3)	E1=Y; k3=k2'; k2=ro; E2=TY; k1=k2
28	X->		degradation	X*k4*(Y+ro*TY)	E1=Y; k2=ro; E2=TY; k1=k4
29	I->II		catalyzed	I*k9*X	E1=X; k=k9
30	II->I		Mass Action	k10*II	Kf=k10

Fig. 8. Budding yeast model in JCMB.

much value to users and some amount of testing for utility should always be made on software that is intended to replace activities in an existing workflow. This is precisely what our first experiment does. We compared the effectiveness of JCMB against the then-existing methodology of one active research group in the field (John Tyson's modeling group). Hand conversion of reaction equations to ODEs is not a strawman comparison, as this was, in fact, the standard procedure for this and other groups for many years and many of the evaluation participants are well-practiced in this process. A second reason for doing such a study is to attempt to identify the remaining deficiencies in the tool so that it can be improved further.

This experiment was performed using five computational cell biology researchers at Virginia Tech. The participants had varying experience levels with symbolic cellular modeling. Some had a small amount of experience with the JCMB tool and one had a background in computer science.

Participants were given two diagrammatic cellular models of medium-difficult complexity, called Model A [3] and Model B [16]. Participants were first given one model and asked to fully represent it as differential equations by hand. Observers watched for critical incidents and mistakes, noting each that was seen. Participants were asked to "think out loud" while working. After participants completed the first model, they were interviewed about their experiences. Of particular interest were critical incidents, signs of confusion, and moments when subjects realized they had been proceeding in an inappropriate direction. If mistakes had been made, subjects were asked to try to find them. They were asked about their normal debugging strategies when in similar situations. Together, researchers and subjects determined whether the debugging strategies suggested would have been effective or successful in locating the particular bug.

After the first model and follow-up questioning were completed, participants were given a second model to represent symbolically, this time using the JCMB environment rather than pencil and paper methods. The follow-up questioning procedure was then repeated. To ensure that observed patterns of errors were related to the method, rather than the model, some subjects used paper and pencil with Model A and JCMB with Model B, while others completed Model B by hand and Model A with JCMB.

Errors were organized on a per-subject basis into four categories: typographical, omission, incomplete, and computational. Typographical errors were a typo or a copying mistake. Omission errors occur when an equation should have been written for a species, but the entire equation was left out. Incomplete errors would fail to identify all of the reactions governing the species' behavior. Computational errors generally involve either incorrect mathematical representation of a term in a differential equation or evidence that the diagrammatic model was incompletely understood.

4.1.1 Paper and Pencil Mistakes

A breakdown of the overall mistake rate and the relative frequency of each type of mistake is shown in Table 1. The denominator for each type of error is the total number of equations. The most common errors were made when a subject used paper and pencil to generate incomplete differential equations. In these cases, the subject would begin an equation for the appropriate species, but would fail to identify all reactions governing the species behavior. Each reaction manifests itself as a term in the equation. This error generally seemed to be caused by distraction or incomplete search of the wiring diagram. The fact that a complete search is required demonstrates that the differential equation model is too far removed conceptually from the diagram model. Converting to the mathematical model is bound to be error prone.

TABLE 1
Breakdown of Observed Mistakes

	Paper/Pencil	JCMB
Mistake Rate	48.43%	4.67%
Computation Errors	30.88%	0%
Incomplete Equation	37.80%	8.33%
Typographical Errors	8.31%	83.33%
Omission Errors	23.01%	8.33%
Omitted Equations	15.77%	7.14%
Total Errors	51	8

Unfortunately, mistakes such as this are not easily preventable. Such mistakes can be typos (reversing the sign on a term) or they might reflect that the math itself was not understood. Omission errors are almost always attributed to incomplete traversal of the diagram model. The errors (both conversion and omission types) cannot be completely prevented by any automated system because they generally lead to syntactically consistent models (even if not the intended model). Typographical errors are different in that an automated system can identify possible (or even probable) points where a mistake has been made. This is the smallest portion of overall errors for the paper and pencil method, but is the most directly preventable.

4.1.2 JCMB Mistakes

Since JCMB is reaction-centric, it alleviates many of the problems observed in paper and pencil entry. By more closely mimicking the diagram model, less effort is required to generate a symbolic model. This reduced effort pays off in a greater ability to notice errors as they are generated. Simply put, the task of error detection (on the part of the user) becomes one of matching (diagram to spreadsheet) rather than computation (requiring searches, mathematical double-checking, etc.).

The overall mistake rate in JCMB (4.67 percent) was approximately one-tenth of that observed with paper and pencil. This rate represents the average over all subjects of the total number of mistakes divided by total number of equations. Error rates were all computed as the number of errors over the number of equations. Percentages for types of mistakes dropped significantly for every error type except typographical. Furthermore, a single subject, a novice, was responsible for all nontypographical errors generated using JCMB, suggesting that, in most cases, these errors are negligible. At the very least, JCMB significantly

addresses these problems, although there is certainly room for improvement to the JCMB system. It is worth remembering here that none of the subjects had substantial previous experience with JCMB.

The increase in percent of typographical errors for JCMB results from decreasing occurrences of every other type of error. Examination of the raw experimental data shows that the same number of typographical errors (six) occurred in both approaches. While this is a small number, it does demonstrate that problems related to typographical errors are insufficiently addressed by JCMB. Redesign of the system has the potential to address problems of this type.

Reduction in mental effort, which affords closer attention to the modeling task, is responsible for at least some of JCMB's success in error reduction. If steps can be taken to further increase attention to the task, rather than to the system, these errors might be virtually eliminated. Suggestions for such improvements were obtained in interviews with subjects.

4.2 A Comparison of the User Interface Paradigms

Our second study attempts to compare the four interface paradigms on their abilities. There are two fundamental activities that a user of a model building tool will wish to do. The first is to enter the model correctly and efficiently. The second is to review the model. Since, in principle, the interface paradigm is independent of the class of pathway models entered, models should (in principle!) be interchangeable between tools supporting various interfaces. Of course, in practice, different tools do support varying classes of models, but we can ignore this if we select test cases that are supported by all of the tools under study. Also, there is no necessary relationship between model entry and model display. That is, a tool could support one interface paradigm for model entry and another for model display. For example, if it were thought that a wizard approach supported users best for model entry, while a spreadsheet was thought to best support model display, a tool using a wizard approach for model entry could switch to a spreadsheet view for display purposes. In practice, most tools today use the same interface paradigm for both model entry and model display. We can speculate that future tools might support multiple interfaces for each task.

A difficulty is that, while we really wish to study interface paradigms and not compare tools, any empirical study requires us to compare specific implementations of those paradigms. So, it becomes difficult to separate the interface paradigm in principle from its implementation as a tool in practice. Our study attempts to minimize this problem. Our approach was to select one simple model (the frog egg model) for entry into a representative implementation for each interface paradigm. The person doing the data entry (Vass) practiced entering the model into each tool before collecting statistics on that tool. In this way, we attempt to measure the minimum time, mouse clicks, and keystrokes required by an "expert" user of the system (that is, only the minimum time needed to physically enter the model is measured, which might approximate the time required by an expert user of the tool). While it is certainly a confounding factor that one system, or one interface paradigm, might be more "intuitive" for initial model entry than another, at least we can claim to compare a minimum cost for model entry.

TABLE 2
A Comparison of Four User Interface Paradigms

Interface	Time	Keystrokes	Mouse Clicks	Screens
Graphical	12 min	479	96	21
Script	10 min	748	1	1
Spreadsheet	10 min	498	60	3
Wizard	12 min	258	95	18

The four tools selected for study are JCMB for the spreadsheet interface, Jarnac for the scripting interface, JDesigner for the graphical interface, and Gepasi for the wizard interface. Again, we wish to stress that our intention is to compare the interface paradigms rather than specific exemplars of implementations.

Table 2 shows the results for entering the frog egg model into each of the four systems. We counted the total amount of time for data entry and the number of mouse clicks and key strokes done during entry. We see that the total time required for each system is roughly the same, with the spreadsheet and scripting interfaces needing about 10 minutes and the graphical and wizard interfaces needing about 12 minutes. As might be expected, the graphical and wizard interfaces tend to require more mouse clicks, while the spreadsheet and script interfaces tend to require more typing. There is one particular anomaly to be pointed out, which is that Gepasi does not support named constants. Thus, while all other systems include time and keystrokes to type both the name and value for various constants, Gepasi allows/requires the user to type only a value. This is a qualitative loss in the information being expressed, leading to a quantitative reduction in data entry.

The second fundamental task for a model editor is viewing the model. The final column of Table 2 shows the total number of screens that a user needed to view in order to see all data for the frog egg model. We see that the spreadsheet and scripting interfaces display the full contents of the model in a relatively small number of screens, while the graphical and wizard interfaces (at least in these implementations) require the user to walk through many screens. Presumably, this would translate into significantly greater viewing times. Again, we point out that a tool could use different interfaces for model entry and model viewing.

5 Conclusions

JCMB is a part of the JigCell problem solving environment [10], a component of the DARPA BioSPICE project. Models of the cell cycle in frog eggs, fission yeast, and budding yeast have been entered using JCMB. Fig. 2 represents a current working version of a frog egg extract model in JCMB entered by Jason Zwolak with the corresponding diagram appearing in Fig. 1. Fig. 8 represents a budding yeast model in JCMB by Andrea Ciliberto.

The spreadsheet interface is also used in other JigCell components where appropriate. The JigCell RunManager [26] allows the user to specify the information necessary to simulate a collection of mutations on a basic "wild type"

model. Each mutation is defined by a (typically small) set of changes to parameter and initial condition settings for a model. For the budding yeast model, we have approximately 130 such mutations to describe. A spreadsheet proves to be a natural way to view this collection of information. The JigCell comparator [1], [2] allows the user to define metrics for comparison and then to view the results of a simulation for such an ensemble of runs. For each mutation, the corresponding experimental data are described, along with an objective function used to define the quality of the match between experimental data and the simulation output. Those results outside a user-defined tolerance are highlighted. Again, a spreadsheet proves to be a natural presentation method.

The usability studies identified areas for improvement to JCMB. These include support for modularization, commenting, and annotation, a comprehensive copy and paste facility, and support for a protein name-matching feature. A simple feature to match new species names to existing ones should reduce the number of typographical errors. The biggest planned change to JCMB is integration with a graphical interface. The spreadsheet interface and the graphical interface have complementary strengths and weaknesses. The graphical interface is useful for generating an overall picture of the relationships between the various species in the model, but is poor at specifying quantitative information. In contrast, the spreadsheet supports entering quantitative information, but is not good at allowing modelers to grasp the overall vision of the model. Our goal is to provide the two interfaces as alternate, simultaneous views into the model. Each interface would support full editing of the model and changes to the model made in one interface would be reflected in the other. In this way, users should get the best of both interface approaches.

ACKNOWLEDGMENTS

This work was supported by the US National Science Foundation Biocomplexity Program, Grant No. MCB-0083315, US National Insitutes of Health Grant 1 R01 GM64339-01, the US Defense Advanced Research Project Agency, and the US Air Force Research Laboratory, Air Force Materiel command, USAF, under agreement number F30602-02-0572.

REFERENCES

[1] N. Allen, L. Calzone, K. Chen, A. Ciliberto, N. Ramakrishnan, C. Shaffer, J. Sible, J. Tyson, M. Vass, L. Watson, and J. Zwolak, "Modeling Regulatory Networks at Virginia Tech," *OMICS, A J. Integrative Biology,* vol. 7, pp. 285-299, 2003.

[2] N. Allen, C. Shaffer, M. Vass, N. Ramakrishnan, and L. Watson, "Improving the Development Process for Eukaryotic Cell Cycle Models With a Modeling Support Environment," *Simulation,* vol. 79, pp. 674-688, 2003.

[3] A. Asthargiri and D. Lauffenburger, "A Computational Study of Feedback Effects on Signal Dynamics in a Mitogen-Activated Protein Kinase (mapk) Pathway Model," *Biotectnology Programming,* vol. 17, pp. 227-239, 2001.

[4] U. Bhalla, "Use of Kinetikit and Genesis for Modeling Signaling Pathways," *Methods Enzymology,* vol. 345, pp. 3-23, 2002.

[5] K. Chen, L. Calzone, A. Csikasz-Nagy, F. Cross, B. Novak, and J. Tyson, "Integrative Analysis of Cell Cycle Control in Budding Yeast," *Molecular Biology of the Cell,* vol. 15, pp. 3841-3862, 2004.

[6] DARPA BioSPICE Web site, https://community.biospice.org, 2005.

[7] DOE, US Dept. of Energy Genomes to Life Web site, http://doegenomestolife.org/, 2005.

[8] M. Ginkel, A. Kremling, T. Nutsch, R. Rehner, and E. Gilles, "Modular Modeling of Cellular Systems with ProMoT/DIVA," *Bioinformatics,* vol. 19, no. 9, pp. 1169-1176, 2003.

[9] 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.

[10] JigCell project Web site, http://jigcell.biol.vt.edu, 2005.

[11] K. Kohn, "Molecular Interaction Map of the Mammalian Cell Cycle Control and DNA Repair Systems," *Molecular Biology of the Cell,* vol. 10, pp. 2703-2734, 1999.

[12] L.M. Loew and J.C. Schaff, "The Virtual Cell: A Software Environment for Computational Cell Biology," *Trends in Biotechnology,* vol. 19, pp. 401-406, 2001.

[13] G. Marlovits, C. Tyson, B. Novak, and J. Tyson, "Modeling M-Phase Control in Xenopus Oocyte Extracts: The Surveillance Mechanism for Unreplicated DNA," *Biophysical Chemistry,* vol. 72, pp. 169-184, 1998.

[14] P. Mendes, "Gepasi: A Software Package for Modeling the Dynamics, Steady States and Control of Biochemical and Other Systems," *Computational Applied Bioscience,* vol. 9, pp. 563-571, 1993.

[15] P. Mendes, "Biochemistry by Numbers: Simulation of Biochemical Pathways with Gepasi 3," *Trends in Biochemical Sciences,* vol. 22, pp. 361-363, 1997.

[16] B. Novák and J. Tyson, "Modeling the Control of DNA Replication in Fission Yeast," *Proc. Nat'l Academy of Science USA,* vol. 94, pp. 9157-9162, 1997.

[17] K. Radmakrishnan and A. Hindmarsh, "Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations," Technical Report URCL-ID-113855, Lawrence Livermore Nat'l Laboratory, 1993.

[18] C. Reder, "Metabolic Control Theory: A Structural Approach," *J. Theoretical Biology,* vol. 145, pp. 175-201, 1988.

[19] H. Sauro, "Jarnac: A System for Interactive Metabolic Analysis," *Animating the Cellular Map: Proc. Ninth Int'l Meeting on BioThermoKinetics,* 2000.

[20] H. Sauro and D. Fell, "Scamp: A Metabolic Simulator and Control Analysis Program," *Math. Computer Modeling,* vol. 15, pp. 15-28, 1991.

[21] H. Sauro, M. Hucka, A. Finney, C. Wellock, H. Bolouri, J. Doyle, and H. Kitano, "Next Generation Simulation Tools: The Systems Biology Workbench ann Biospice Integration," *OMICS: A J. Integrative Biology,* vol. 7, pp. 355-372, 2003.

[22] H. Sauro and E. Ingalls, "Conservation Analysis in Biochemical Networks: Computational Issues for Software Writers," *Biophysical Chemistry,* vol. 109, pp. 1-15, 2004.

[23] H. Sauro, J. Small, and D. Fell, "Metabolic Control and Its Analysis. Extensions to the Theory and Matrix Method," *European J. Biochemistry,* vol. 175, pp. 216-221, 1987.

[24] J. Schaff and L. Loew, "The Virtual Cell," *Proc. Pacific Symp. Biocomputing,* pp. 228-239, 1999.

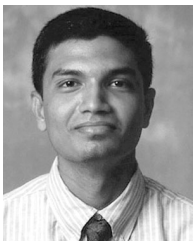
[25] K. Takahashi, N. Ishikawa, Y. Sadamoto, H. Sasamoto, S. Ohta, A. Shiozawa, F. Miyoshi, Y. Naito, Y. Nakayama, and M. Tomita, "E-Cell 2: Multiplatform E-Cell Simulation System," *Bioinformatics,* vol. 19, no. 13, pp. 1727-1729, 2003.

[26] M. Vass, N. Allen, C. Shaffer, N. Ramakrishnan, L. Watson, and J. Tyson, "The Jigcell Model Builder and Run Manager," *Bioinformatics,* vol. 20, pp. 3680-3681, 2004.

[27] M. Vass and P. Schoenhoff, "Error Detection Support in a Cellular Modeling End-User Programming Environment," *Proc. IEEE Symp. Human Centric Compter Language and Environments,* pp. 104-106, 2002.

Marc T. Vass is a PhD candidate in the Department of Computer Science at Virginia Tech. He received the BS degree in computer science from Virginia Tech in 2000 and the MS degree in computer science from Virginia Tech in 2001. His research interests include creativity and cognition, problem solving environments, flow, and theoretical human computer interaction.

Clifford A. Shaffer received the PhD degree from the University of Maryland in 1986. He is an associate professor in the Department of Computer Science at Virginia Tech. His current research interests include problem solving environments, bioinformatics, component architectures, visualization, algorithm design and analysis, and data structures. He is a member of the IEEE and the IEEE Computer Society.



Naren Ramakrishnan received the PhD degree in computer sciences from Purdue University in August 1997. He is an associate professor of computer science and a faculty fellow at Virginia Tech. He also serves as an adjunct professor at the Institute of Bioinformatics and Applied Biotechnology (IBAB), Bangalore, India. His research interests are computational science, problem solving environments, mining scientific data, and information personalization. He currently serves on the editorial board of *Computer*. He is a member of the IEEE Computer Society.

Layne T. Watson received the BA degree (magna cum laude) in psychology and mathematics from the University of Evansville, Indiana, and the PhD degree in mathematics from the University of Michigan, Ann Arbor. He is a professor of computer science and mathematics at Virginia Tech. His research interests include fluid dynamics, structural mechanics, homotopy algorithms, parallel computation, mathematical software, and image processing. He has worked for USNAD Crane, Sandia National Laboratories, and General Motors Research Laboratories and served on the faculties of the University of Michigan and Michigan State University, East Lansing, before coming to Virginia Tech. He is a fellow of the IEEE and a member of the IEEE Computer Society.

John J. Tyson received the PhD degree in chemical physics from the University of Chicago in 1973 and has been specializing in theoretical cell biology since that time. He is University Distinguished Professor of Biological Sciences at Virginia Tech. His current interests revolve around the gene-protein interaction networks that regulate features of cell physiology such as cell division, circadian rhythms, intracellular signaling networks, and programmed cell death.