Dizzy: a software system for simulating the chemical kinetics of complex systems

Stephen Ramsey[*Δ] and Hamid Bolouri[*]
October 23, 2003

Many steps in gene expression involve small numbers of molecules, such as two copies of the gene or small numbers of RNA molecules being transcribed. As a result gene expression patterns between cells can be highly variable, as demonstrated experimentally by Elowitz *et al*.[1] and others. To date, it has been difficult to study the degree to which such stochastic effects influence the behavior of regulatory networks in individual cells. This difficulty is because it has been computationally expensive to conduct stochastic simulations except in the simplest systems such as the lysis/lysogeny bifurcation in the life cycle of phage lambda.

Dizzy[2] is a modeling and simulation environment optimized for the study of possible stochastic effects in genetic and protein interaction networks. In Dizzy, chemical reaction kinetics may be modeled deterministically using ordinary differential equations (ODEs), or as a stochastic process using Monte Carlo techniques. The usefulness of stochastic simulations for modeling biochemical processes is well documented in the literature.[3,4,5,6] Dizzy is intended to facilitate detailed quantitative modeling of the dynamics of complex biochemical systems from both a deterministic and stochastic viewpoint. In Dizzy, one can switch between a deterministic and a stochastic simulation with a single click of the mouse button, without needing to make any changes to the model. The Dizzy system includes a graphical user interface, a model definition language ("Dizzy"), model import/export of Systems Biology Markup Language[7] (SBML), and the ability to plot simulation results or save simulation results in a comma-delimited file. Dizzy is implemented entirely in the Java™ programming language[8].

---

* Institute for Systems Biology, Seattle, Washington, USA.
Δ To whom correspondence should be addressed: sramsey@systemsbiology.org.

1 Michael B. Elowitz et al. "Stochastic Gene Expression in a Single Cell." *Science* 297 (2000);1183-1186.
2 Ramsey, Stephen. "Dizzy Home Page." Institute for Systems Biology. Oct. 21, 2003. http://labs.systemsbiology.net/bolouri/software/Dizzy
3 Ackers, Gary K., Alexander D. Johnson, and Madeline A. Shea. "Quantitative model for gene regulation by lambda phage repressor." *Proc. Natl. Acad. Sci.* 79 (1982):1129-1133.
4 Adam Arkin, John Ross, and Harley H. McAdams. "Stochastic Kinetic Analysis of Developmental Pathway Bifurcation in Phage lambda-Infected *Escherichia coli* Cells." Genetics 149 (1998):1633-1648.
5 Michael Andrew Gibson and Jehoshua Bruck. "A probabilistic model of a prokaryotic gene and its regulation." California Institute of Technology, Department of Computation and Neural Systems Preprint 136-93. (1999).
6 Orrell, David. "Stochasticity and stability in biochemical systems." In preparation. (2003).
7 Hucka, Mike, *et al*., "The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models." *Bioinformatics* 19 (2003):524-531.
8 Gosling, James, *et al.*, "The Java Language Specification." 2nd ed. New York: Addison-Wesley, 2000. http://java.sun.com/docs/books/jls/second_edition/html/j.title.doc.html

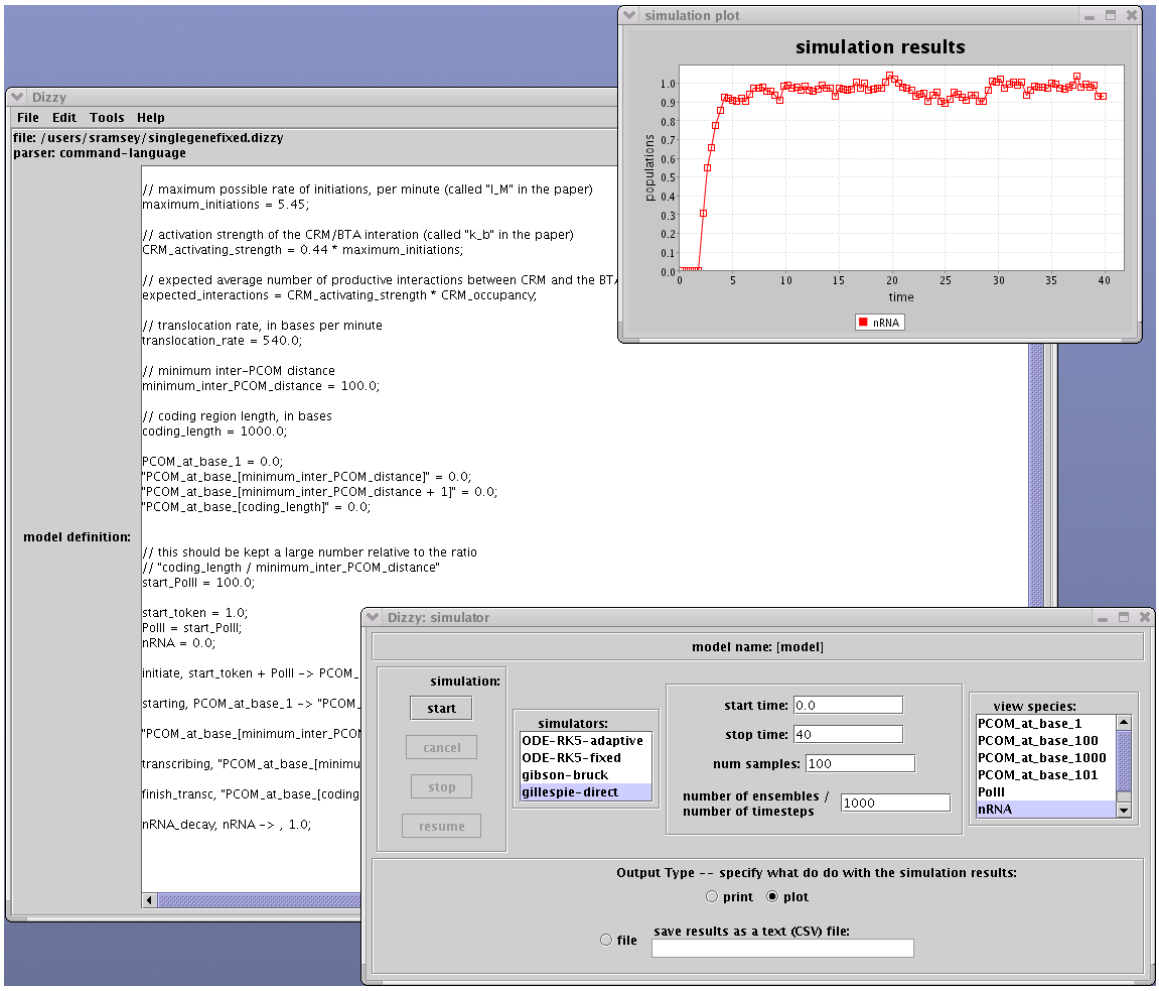

Illustration 1: screen shot of the Dizzy software system

Deterministic modeling is accomplished using an ODE representation of the system of chemical reactions. Dizzy provides two different implementations of ODE solvers, a 4th-order Runge-Kutta solver with fixed step-size[9] (which can be specified by the user), and a 5th-order Runge-Kutta solver with adaptive step-size control[10]. The deterministic simulators are capable of modeling the dynamics of chemical reactions with ordinary kinetics (product of reactant concentrations and kinetic constant) or custom, user-defined rate expressions.

9 Stoer, J., and R. Bulirsch. *Introduction to Numerical Analysis.* 2nd ed. Heidelberg: Springer-Verlag, 1993.

10 Press, William, *et al. Numerical Recipes in C*. Cambridge: Cambridge University Press, 1988.

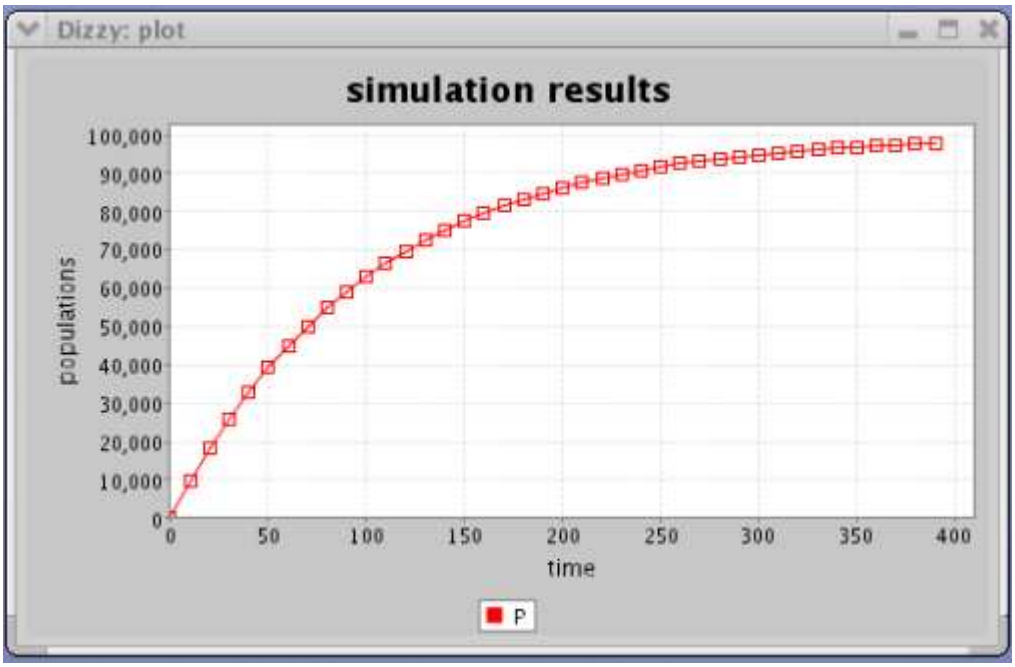

Illustration 2: deterministic simulation of Michaelis-Menten kinetics

Reaction kinetics may be modeled as a stochastic process, employing Monte Carlo simulation techniques. Dizzy includes two different stochastic simulators: an implementation of the "direct method" of the Gillespie[11] algorithm, and an implementation of the "next reaction method" of the Gibson-Bruck[12] algorithm. In Dizzy, stochastic simulations may be used to model the kinetics of chemical reactions with ordinary rate laws, or (with some amount of computational overhead) custom rate expressions. An ensemble of realizations of the stochastic process are conducted, and the results of the simulation are computed as averages over the ensemble, of the desired species populations, at each time. The ensemble size is specified by the user.

11 Gillespie, D. T. "A General Method for Numerically Simulating the Stochastic Evolution of Coupled Chemical Reactions." *J. Comp. Phys.* 22 (1976):403-434.

12 Gibson, Michael A., and Jehoshua Bruck. "Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels." California Institute of Technology preprint ETR026 (1998)

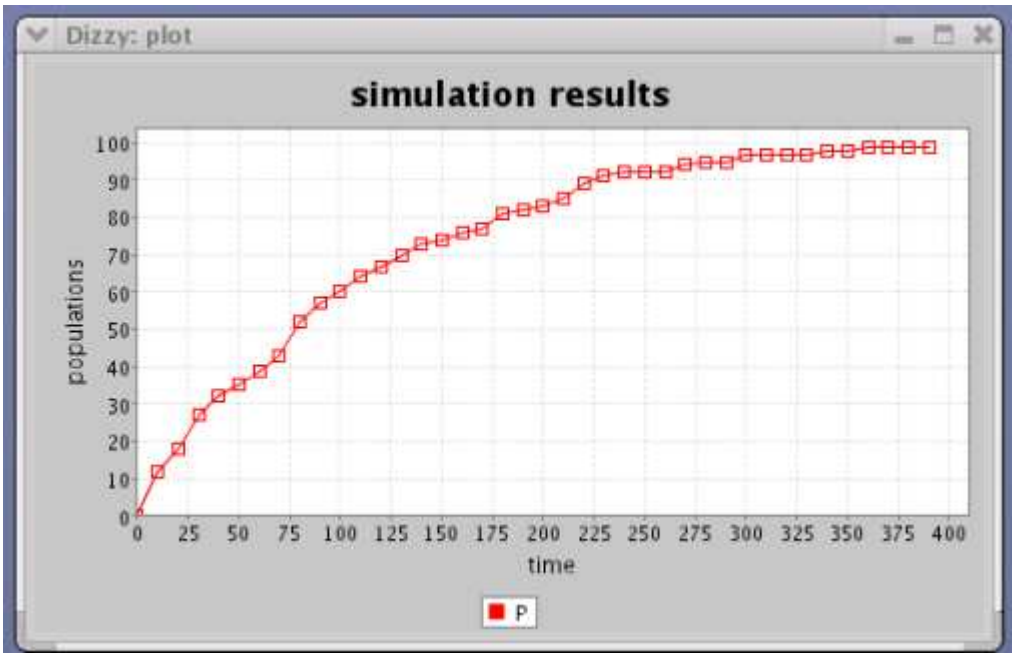

Illustration 3: Stochastic simulation of Michaelis-Menten kinetics, showing noise effects

Dizzy has support for defining symbols, and composing symbols and simple arithmetic operators and transcendental functions into mathematical expressions. A symbol may be associated with a mathematical expression for immediate evaluation at the time of parsing (the default), or for deferred evaluation in the simulator (using square brackets). Deferred-evaluation mathematical expressions may also involve the special symbol "time", which represents the elapsed time of the simulation. A symbol may be promoted to be a species, compartment, or reaction. In the case of a species, the associated value of (or expression for) the symbol represents the species population. In the case of a compartment, the associated value of (or expression for) the symbol represents the compartment volume. In the case of a reaction, the associated value (or expression for) the symbol represents the method used to compute the rate of the reaction.

Species symbol values ordinarily represent the initial population (or concentration) of a chemical species. A species may participate in a chemical reaction as a "boundary" species, in which case its population (concentration) is unaffected by the rate of the given reaction. A species symbol defined as a mathematical expression may only participate as a boundary species in any chemical reaction; its value is always determined by evaluating the mathematical expression.

Reaction rates are computed in one of two ways. If the reaction symbol is a simple value, the reaction rate is computed using the product of the symbol value and the populations (or concentrations) of all reactant species. If the reaction symbol is a mathematical expression, the reaction rate is instead computed by evaluating the expression. This permits the user to compose custom rate laws involving any symbols in the model.

The Dizzy simulation controller is used to initiate simulations. A variable number of

species may be selected to "view," and time-series data for these species will be the output of the simulation. In addition, the start and stop times for the simulation are specified, as well as the simulation method to use (Gibson, Gillespie, ODE, *etc.*).

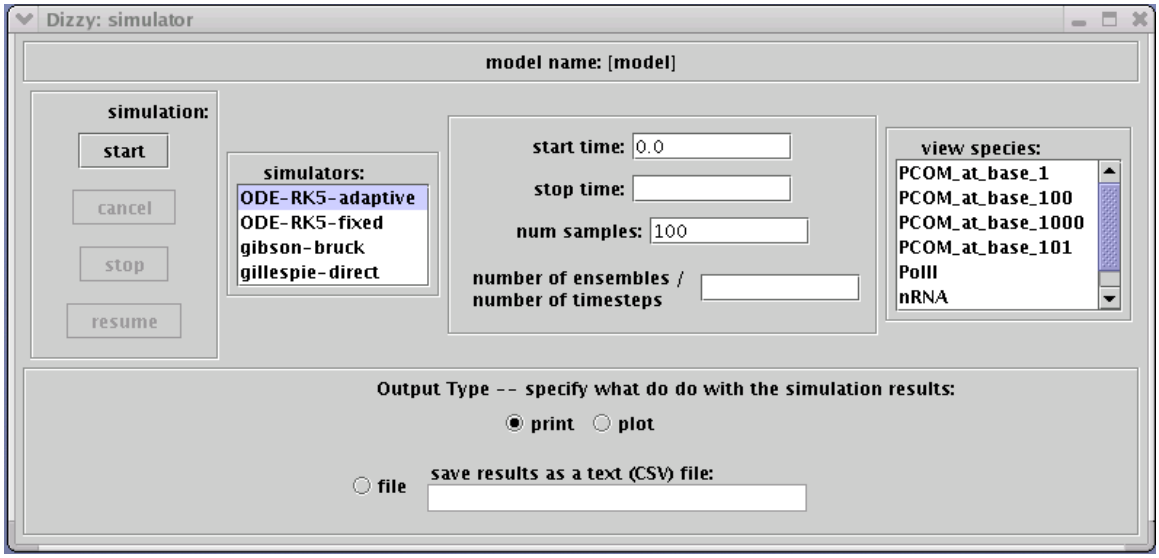

Illustration 4: Dizzy simulation controller screen

Model definition is accomplished using an SBML Level 1 import feature, or using the "Dizzy" language. The SBML import feature is capable of parsing models conforming to either Version1 or Version 2 of the SBML Level 1 specification[13]. The Dizzy language is a simple declarative language for defining species, reactions, and initial conditions. The design goals of the Dizzy language were simplicity and intuitiveness. The language has basic support for mathematical expressions, variables, looping constructs, and nested file inclusion. A simple example of a model defined in the Dizzy language is shown here:

```
k1 = 1.0;      // kinetic parameter for reaction1
A = 100.0;     // initial amount of species A
B = 100.0;     // initial amount of species B
C = 0.0;       // initial amount of species C
D = 0.0;       // initial amount of species D
               // the following is a simple reaction:
reaction1, A + B -> C + D, k1;
               // here we use a custom rate expression:
reaction2, C -> B, [ k2 * C * time ];
```

The Dizzy language syntax has some similarities with the Jarnac[14] modeling environment, as well as some syntactic features borrowed from the C programming

13 Hucka, Michael, et al. "Systems Biology Markup Language (SBML) Level 1: Structures and Facilities for Basic Model Definitions." Systems Biology Workbench Development Group and JST ERATO Kitano Symbiotic Systems Project. August 28, 2003. `http://www.sbml.org/documents`.

14 Sauro, Herbert. *Jarnac (Scamp II) – Reference Guide.* Vers. 1.19. California Institute of Technology, Nov. 15, 2001. `http://www.cds.caltech.edu/~hsauro/docs/JRef.pdf`.

language[15]. Dizzy is capable of exporting a model that was initially defined using the Dizzy language, to an SBML Level 1 representation of the model. An example of a model imported from SBML (the "Repressilator" model of Elowitz *et al.*[16]) and simulated in Dizzy is shown below:

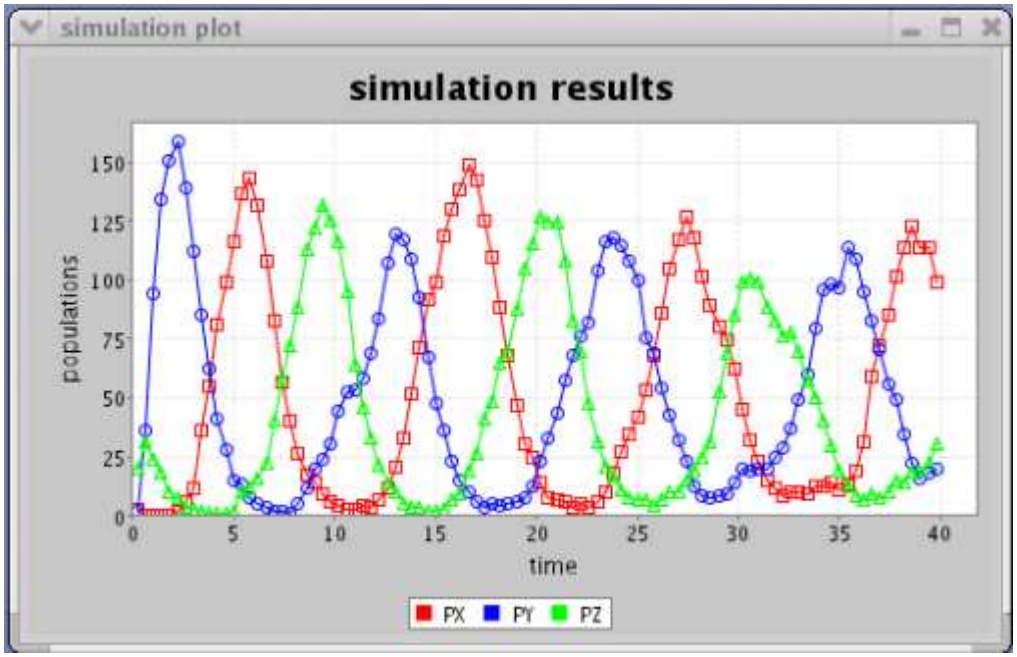

Illustration 5: stochastic simulation of the "Repressilator" model of Elowitz et al.*, imported from SBML (ensemble average over 10 realizations of the stochastic process)*

Dizzy has an experimental feature in which a "multi-step" chemical reaction may be defined[17]. This is used for simulating the kinetics of a long chain of reaction steps with equal rate constants. This feature is implemented using a delay function in the deterministic simulators, and using a priority queue in the stochastic simulators. This feature may prove useful for simulating many-step processes such as transcription.

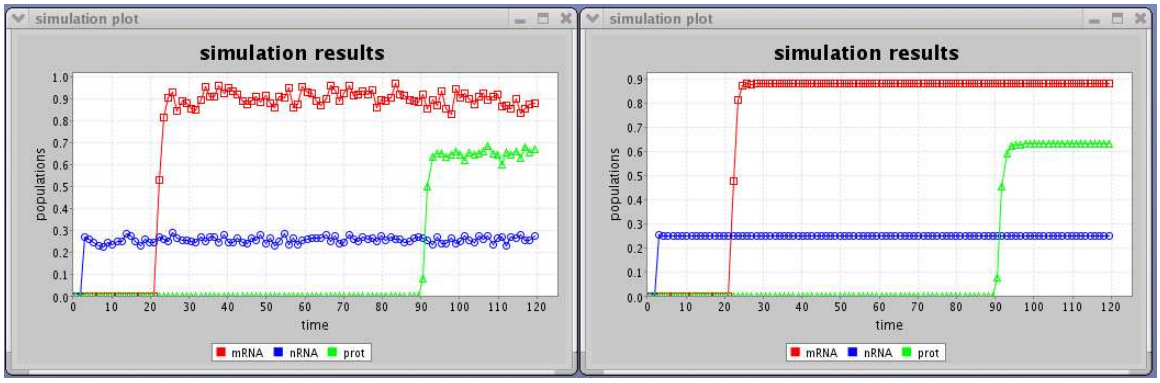

Illustration 6: stochastic and deterministic simulations of a model involving a many-step reaction chain

15 Kernighan, Brian W., and Dennis M. Ritchie. *The C Programming Language*. 2nd ed. New York: Prentice-Hall, 1988.

16 Elowitz, M. B. *et al.* "A Synthetic Oscillatory Network of Transcriptional Regulators." *Nature* 403 (2000):335-338.

17 *ibid.* Gibson and Bruck, 1998.

Dizzy is able to interoperate with the Systems Biology Workbench[18] (SBW) software integration environment. It offers the Gillespie simulator as a simulation service conforming to the SBW Simulation Service interface specification[19]. In this manner, the Dizzy Gillespie simulator may be remotely invoked using the SBW system. Dizzy provides a program that may be used to register the Gillespie simulator with a running SBW Broker.

Dizzy provides an optional feature where a user may display a graphical representation of a model, using a he Cytoscape software product[20]. The model is exported into SBML, and loaded into a customized SBML-enabled version[21] of Cytoscape. The special version of Cytoscape is invoked remotely from a server based at Institute for Systems Biology, using the Java™ Network Loading Protocol[22] (JNLP). The necessary Cytoscape library is automatically downloaded and cached locally, and it is (if necessary) updated transparently using JNLP whenever the "view model" feature is activated.

---

18 Hucka, Michael, *et al.* "Introduction to the Systems Biology Workbench." Systems Biology Workbench Development Group and JST ERATO Kitano Symbiotic Systems Project. Aug. 8, 2003. http://www.sbw-sbml.org/sbw/docs/intro/intro.pdf.

19 Finney, Andrew, et al. "Systems Biology Workbench Module Programmer's Manual." ERATO Project. May 31, 2002.
http://www.sbw-sbml.org/sbw/docs/module-api/module-api.pdf.

20 Ideker, T., et al. Cytoscape. Vers. 1.1. Computer software. Cytoscape Consortium, 2003. http://www.cytoscape.org.

21 Shannon, Paul. Cytoscape SBML Reader Plug-in. Vers. 1.0. Computer Software. Institute for Systems Biology, Oct. 3, 2003. http://www.cytoscape.org/plugins.

22 Schmidt, René. "Java™ Network Launching Protocol & API Specification (JSR-56)." Vers 1.0.1. Sun Microsystems, May 21, 2001.
http://java.sun.com/products/javawebstart/download-spec.html

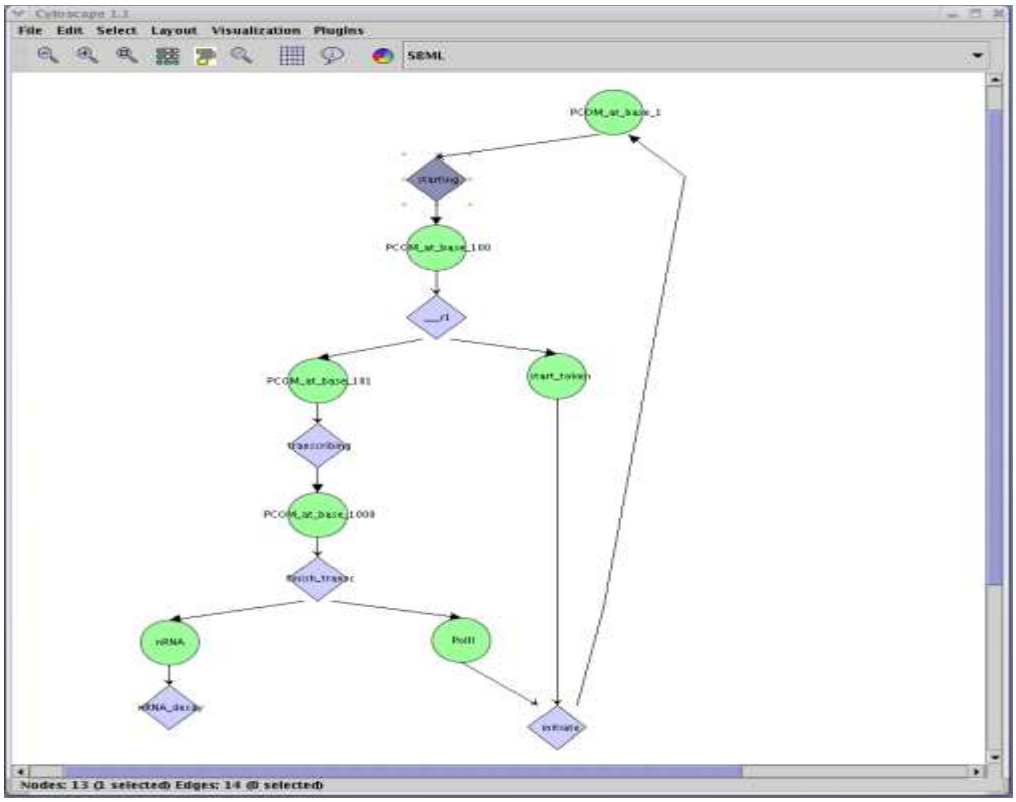

Illustration 7: View of a Dizzy model in Cytoscape, courtesy of Paul Shannon (ISB)

Dizzy is an open source software product. It is available free of charge to the scientific community. It is released under the GNU Lesser General Public License[23] (LGPL). This license permits (in principle) using the Dizzy library to produce closed-source software products, subject to the limitation that the Dizzy source code must be included with any distribution of derivative work. No legal encumbrances whatsoever are placed upon discoveries or publications based upon results obtained using the Dizzy simulator.

Dizzy is a well-documented software system. A comprehensive user manual[24] is available on the web in both Hypertext Markup Language[25] (HTML) and Portable Document Format[26] (PDF) form. There is a graphical help-browsing feature within the Dizzy program using the Sun JavaHelp[27] feature. The application programming interface

---

23 Stallman, Richard, et al. "GNU Lesser General Public License." Vers. 2.1. Free Software Foundation, Feb. 1999. `http://www.gnu.org/copyleft/lesser.html`.

24 Ramsey, Stephen. "Dizzy User Manual." Vers. 0.0.6. Institute for Systems Biology, Oct. 21, 2003. `http://labs.systemsbiology.net/bolouri/software/Dizzy/docs/UserManual.pdf`.

25 Steven Pemberton *et al*. "XHTML ™ 1.0 The Extensible HyperText Markup Language." 2nd Ed. World Wide Web Consortium. Aug. 1, 2002. `http://www.w3.rg/TR/20-02/REC-xhtml1-20030801`

26 Adobe Systems Incorporated. "PDF Reference: Adobe portable document format version 1.4" 3rd Ed. New York: Addison-Wesley, 2003. Adobe Systems Incorporated. Nov. 2001. `http://partners.adobe.com/asn/tech/pdf/specifications.jsp`

27 Brinkley, Roger. "JavaHelp 2.0 Specification (JSR-097)." Sun Microsystems. Mar. 13, 2003. `http://java.sun.com/products/javahelp/download_spec.html`

(API) for Dizzy is documented on the web using JavaDoc[28], and it is available in both HTML and PDF formats. To ensure consistency, all forms of the Dizzy user manual (HTML, PDF, and JavaHelp) are generated from a common extensible markup language[29] (XML) source file via the "extensible stylesheet language" (XSL) using the Apache Xalan[30] XSL compiler. A dedicated electronic mail alias has been established for correspondence related to Dizzy, and specifically for user inquiries and bug reports: dizzy@systemsbiology.org.

Dizzy is easy to download and install. On most platforms, the program may be downloaded as a single executable file that contains all software needed in order to run the program. The Dizzy download may optionally include the Java™ Runtime Environment (JRE). The executable that is downloaded is a user-friendly installation manager program that installs Dizzy and its configuration files and libraries, and (optionally) creates symbolic links so that Dizzy is visible on the desktop (or the Microsoft Windows™ "Start Menu"). The installation manager is created using the InstallAnywhere product.[31] When downloaded without the Java Runtime Environment, the Dizzy installer is a mere 3.3 megabytes (MB). The "one click" installation manager program is available for Windows and Linux, and (with a command-line invocation required) for any other Java-enabled operating system. A "one click" installer for the Apple Macintosh™ operating system version 10 ("Mac OSX") is likely to be available in the near future.

Dizzy's installed (and runtime) footprint is relatively small. It requires only 3.7 MB of disk space, when installed without Java Runtime Environment (not including the size of the Cytoscape library that gets stored in the JNLP cache). The Cytoscape library that is optionally downloaded via JNLP is 3.2 MB. When running, Dizzy uses anywhere from 50-100MB of RAM, depending on the complexity of the model and the type of Java Runtime Environment used. The standard Dizzy installation includes all libraries required for Dizzy to function, including a JNLP library that is used to access Cytoscape over the web.

Dizzy is a Java-based software system. It is written entirely in Java, and uses no "native" (architecture-specific) libraries, other than the standard libraries that ship with the Java Runtime Environment. The graphical user interface feature is provided using the Swing user interface components of the Java™ Foundation Classes (JFC). These user interface components have a relatively uniform "look-and-feel" across different operating system platforms. On-line help browsing is provided using the Sun JavaHelp feature. The build system used for Dizzy is the Apache Ant system, which permits the building of Dizzy on

28 Sun Microsystems Inc. "Javadoc Tool Home Page." Sun Microsystems Inc. Oct. 2003. http://java.sun.com/j2se/javadoc

29 Tim Bray *et al.* "Extensible Markup Language 1.0." 2nd Ed. World Wide Web Consortium. Oct. 6, 2000. http://www.w3.org/TR/2000/REC-xml-20001006

30 Apache XML Project. "Xalan-Java version 2.5.1." Apache Foundation. Oct. 2003. http://xml.apache.org/xalan-j

31 Zero-G Software. InstallAnywhere. Vers. 5. 0.1. Computer program. http://www.zerog.com

any Java-enabled platform. The parsing and writing of XML files is accomplished using the Apache Crimson[32] XML library. The "one-click" installer for Dizzy is built using InstallAnywhere, a Java- and XML-based product.

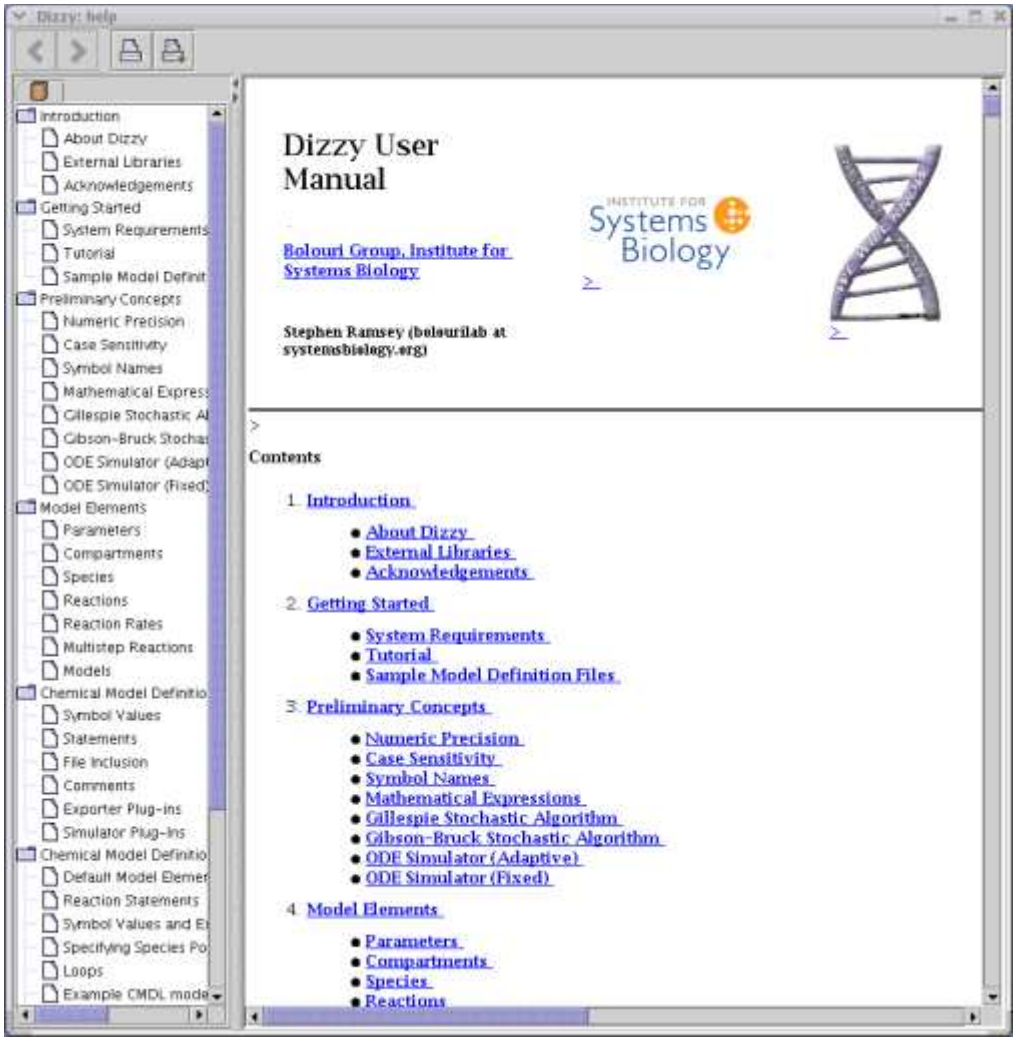

Illustration 8: screen shot of the Dizzy help screen

Dizzy is presently under active development. Efforts are underway to add support for modeling diffusion and spatially inhomogeneous species concentrations, as well as a template feature to facilitate model element re-use. In addition, extensions to the Dizzy language to support hierarchical model-building are being explored. The Dizzy system is currently being used to model the dynamics of transcriptional regulatory processes in the developing *Stronylocentrotus purpuratus* (sea urchin) embryo[33].

32 Apache XML Project. "Crimson 1.1 Release" Apache Foundation. Oct. 2003. `http://xml.apache.org/crimson`

33 H. Bolouri and E. H. Davidson, "Transcriptional regulatory cascades in development: Initial rates, not steady state, determine expression kinetics." Available FRPNAS early edition (2003).