The Systems Biology Workbench (SBW) Version 1.0: Framework and Modules

Andrew Finney[1,2,3], Michael Hucka[1,2], Herbert Sauro[1,2], Hamid Bolouri[1,2,3], Akira Funahashi[2], Ben Bornstein[1,2,5], Ben Kovitz[1,2], Joanne Matthews[3], Bruce Shapiro[1,2,5], Sarah Keating[3], John Doyle[1,2], Hiroaki Kitano[1,2,4,6]

[1]California Institute of Technology, Pasadena, CA, USA
[2]JST/ERATO Kitano Symbiotic Systems Project, Tokyo, Japan
[3]University of Hertfordshire, UK
[4]The Systems Biology Institute, Tokyo, Japan
[5]NASA Jet Propulsion Lab, Pasadena, CA, USA
[6]Sony Computer Science Labs, Tokyo, Japan

Introduction

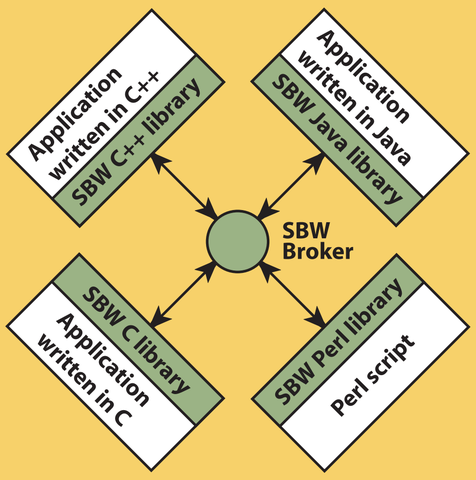
Progress in molecular biotechnology has fueled an explosion in the development of software tools. Regrettably, developers often end up recreating similar facilities in separate software packages.

In an effort to make it more attractive for developers to share rather than reimplement resources, we have implemented the Systems Biology Workbench (SBW), a free, **open-source, application integration environment**. Our goal has been to create a framework simple enough that software authors find it easier to provide an SBW interface than to recreate functionality available in other tools. By doing so, we hope **developers can concentrate on creating best-of-breed solutions** in their areas of expertise.

What Does SBW Provide?

SBW provides libraries for enabling applications to learn about and communicate with each other. The applications may be running on separate computers.

SBW lets heterogevneous packages connect to each other using a **remote procedure call** mechanism based on a message-passing network protocol. The interfaces to SBW are encapsulated in client libraries for different languages.



The *SBW Broker* starts applications on demand, and coordinates communications on a given computer.

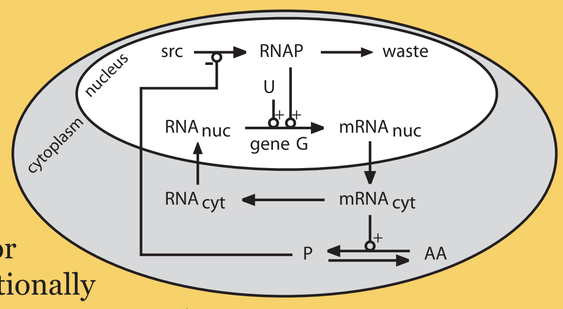
A Broker is started automatically for the user if one is not running when the first SBW application starts.

Features of SBW Version 1.0

- Languages supported: **C**, **C++**, **Delphi**, **Java**, **Perl**, and **Python**.
- **Windows (98, 2000, XP)** and **Linux** supported, with **MacOS X** planned in the near future.
- **Secure, distributed operation via SSH**, featuring remote startup of brokers and applications.
- **CORBA gateway** for bidirectional communication between SBW-based apps and CORBA-based apps.
- **Collection of basic applications** provided with the SBW distribution, including:
  - A simple **stochastic simulator** based on the Gibson-Bruck variant of the Gillespie algorithm
  - An **SBML-to-MATLAB** ODE & Simulink translator
  - An **SBML reader tool** that allows a program to extract (via an API) components of an SBML model
  - A **"clipboard" module** that stores an SBML model description, and allows the easy transfer of models between separate modules
  - A **"browser" module** that allows querying SBW for registered modules and producing descriptions of each module's interface in Java or CORBA IDL
  - A simple **plotting module** for time-series data
  - A **generic simulation control GUI** interface.
  - A collection of tutorial **example modules** in C, C++, Delphi and Java
- **Extensive documentation**—in addition to overview documents and published papers, every language library has its own programmer's manual and API reference.

SBW in Action: A Sample Session

Here is an example of using several SBW-enabled tools to create and simulate a two-compartment model of a hypothetical single-gene oscillatory circuit.

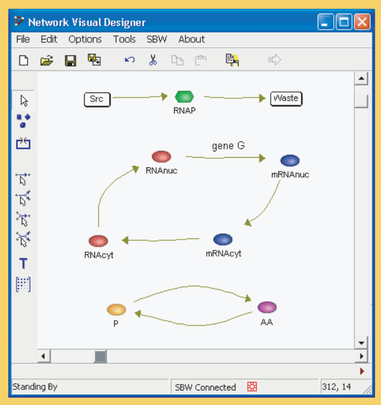


In this highly simplified model, there is a gene G which encodes its own repressor and is transcriptionally activated at a constant rate, *Vi*. Transcriptional activation of a gene G (which normally involves many enzymatic reactions) is summarized here as the production of active *RNAP* from source material, *src*, and degradation to *waste*. Transcribed *mRNA* is then transported from the nucleus into the cytoplasm, where it is translated into the product *P* from constituent amino acids *AA* and where it is also subject to degradation.

Reaction	Rate
$src \rightarrow RNAP$	$\frac{V_i}{1 + P/K_i}$
$RNAP \rightarrow waste$	$V_{kd} \cdot RNAP$
$RNA_{nuc} \rightarrow mRNA_{nuc}$	$\frac{V_{m1} \cdot RNAP \cdot RNA_{nuc}}{K_{m1} + RNA_{nuc}}$
$mRNA_{nuc} \rightarrow mRNA_{cyt}$	$k_1 \cdot mRNA_{nuc}$
$mRNA_{cyt} \rightarrow RNA_{cyt}$	$\frac{V_{m2} \cdot mRNA_{cyt}}{mRNA_{cyt} + K_{m2}}$
$RNA_{cyt} \rightarrow RNA_{nuc}$	$k_2 \cdot RNA_{cyt}$
$AA \rightarrow P$	$\frac{V_{m3} \cdot mRNA_{cyt} \cdot AA}{AA + K_{m3}}$
$P \rightarrow AA$	$\frac{V_{m4} \cdot P}{P + K_{m4}}$

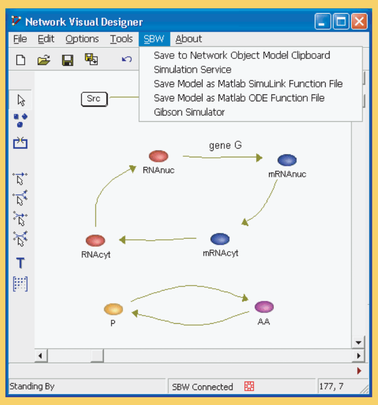
Model Capture: Using a Visual Editor

Using the SBW-enabled *JDesigner* biochemical network editor, a user can create the model using a graphical interface.

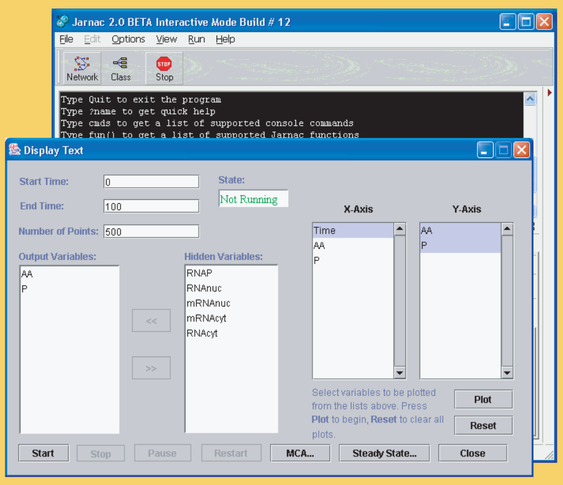


Model Simulation: Exchanging Models via SBW

An application such as JDesigner can dynamically create a menu of tools with which it can interact, by querying SBW to find all installed SBW-enabled packages that provide services for processing biochemical models.

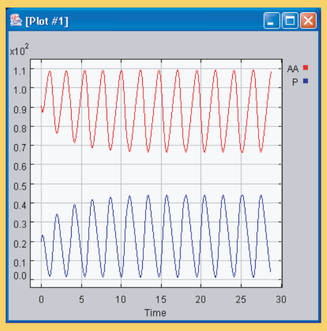


Here, picking the menu item "Simulation Service" invokes a generic simulation control GUI, which in turn invokes *Jarnac*, an ODE-based simulator for biochemical reaction networks.



Model Visualization and Analysis

Setting the run parameters for the simulation and selecting the output variables in the simulation control GUI allows the user to plot the values of quantities over time. The plot at the right shows how the concentrations of *AA* and *P* in the model oscillate over time.



A user can also perform other analyses on the model via SBW, e.g., by invoking the bifurcation analysis module.

Third-Party Modules Available For SBW

- ***Jarnac***, a biochemical simulation package for Windows
- ***JDesigner***, a visual biochemical network layout tool
- ***Pasadena Twain***, a simple interactive ODE solver
- A **stochastic simulator** based on Gillespie's algorithm
- A **bifurcation analysis** module
- An **optimization** module
- An **SBML validator** for checking SBML model files
- An **inspector** that lists running modules & their services

Coming Attractions

More open-source developers are joining the SBW project, and together we are enhancing and extending SBW in many ways. Here is a preview of some coming attractions:

- Support for JDK 1.4
- Support for MacOS X
- MATLAB scripting interface
- New modules, including:
  - Graphical browser for the SBW environment
  - Rewritten SBML parser/writer for SBML Level 2
  - Improved generic GUI for simulators
  - Improved, full-featured plot module
  - New simulation engines

How to Get Started with SBW

The SBW version 1.0 package and extensive documentation are available from the project web site,

http://www.sbw-sbml.org/.

SBW is distributed under the terms of the GNU LGPL.

Acknowledgments

The SBW project is funded by a generous grant from the Japan Science and Technology Corporation under the ERATO Kitano Symbiotic Systems Project.

We thank the following groups engaged in developing software tools for systems biology, for their feedback and guidance in developing SBW:

- BioSpice (Arkin et al.)
- Cellerator (Mjolsness et al.)
- DBsolve (Goryanin et al.)
- E-CELL (Tomita et al.)
- Gepasi (Mendes et al.)
- ProMoT/DIVA (Gilles et al.)
- StochSim (Bray et al.)
- Virtual Cell (Loew et al.)

We also thank the following individuals for their comments, suggestions and assistance: Tau-Mu Yi, Mineo Morohashi, Pieter van der Zee, Fred Livingston, Paul Shannon, Deanne Taylor, Andrew de Laix, Venkat Jagadish, Lukasz Salwinski, Mark Johnson, Richard Giuli, Greg Riddick, Michael Vanier.

open source
SOURCEFORGE.net