Overview

- Introduction of CellDesigner
- SBML (Systems Biology Markup Language)
- SBGN (Graphical Notation)

How to build a model with CellDesigner
How to create CellDesigner plugin
Software Infrastructure

Model representation
Standard representation method of biological models

Database

Software tools

CellDesigner

Jarnac
Plot
Gibson
CellDesigner

Modeling tool for biochemical and gene-regulatory network
SBML (Systems Biology Markup Language)

A machine-readable format (XML) for representing computational models in systems biology.
Reactions According to SBML

'Kinetic law':
\[ v = f(R, P, M, \text{parameters}) \]
What does SBML look like?

Biochemical reaction

\[ k \cdot [S1] \]

```xml
<listOfSpecies>
  <species id="s1" name="s1" compartment="default"
initialAmount="0" charge="0"/>
  <species id="s2" name="s2" compartment="default"
initialAmount="0" charge="0"/>
</listOfSpecies>

<listOfReactions>
  <reaction id="re1" reversible="false" fast="false">
    <listOfReactants>
      <speciesReference species="s1"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="s2"/>
    </listOfProducts>
    <kineticLaw formula="k*s1"/>
  </reaction>
</listOfReactions>
```
Over 110 software packages support SBML

http://sbml.org
SBGN

Graphical Notation for representing biological interactions

protein–protein interaction, gene regulatory networks

SBGN community

- BioModels Database (UK)
- BioNetGen (USA)
- BioPAX
- BioUML (Russia)
- CellDesigner (Japan)
- CellML (New Zealand)
- COPASI (Germany)
- Cytoscape (USA)
- Design Suite (USA)
- EPE, EPN (UK)
- INOH (Japan)
- JDesigner (USA)
- Narrator (UK)
- NetBuilder
- Panther (USA)
- ProcessDB
- ProMot (Germany)
- QBT (USA)
- SABIO-RK (Germany)
- SBML Layout extension
- Taverna (UK)
- VCell (USA)

And more...
SBGN Level-1

Graphical Notation for representing biological interactions

protein–protein interaction, gene regulatory networks
Notation

Syntax for index on category-II reduced notation

EffectDescription = Result ImmediateEffect Condition | SimpleSentence

Result = TRANSITION (+ | `-` | empty) `<-` | empty
ImmediateEffect = TERM_IE
Condition = empty | `('TERM_COND')`
SimpleSentence = `('+` | `-` | `=' TRANSITION `|` ?`

TERM_IE = RESIDUE_IE | TERM_IE OP TERM_IE
TERM_COND = RESIDUE_COND | TERM_COND OP TERM_COND
RESIDUE_IE = `('+` | `-`) MODIFICATION (`@` TYPE [0-9]+ SUBUNIT | empty)
RESIDUE_COND = `('-' | `=` empty) MODIFICATION `@` TYPE [0-9]+ OP
MODIFICATION = (P | Me | Ac | Ub | Hy) | (P | M | A | U | H)
TYPE = (Tyr | Ser | Thr) | (Y | S | T)
SUBUNIT = empty | `'/` SUBUNIT_NAME
TRANSITION = [a-zA-Z]+[0-9]*
SUBUNIT_NAME = [a-zA-Z]+[0-9]*

Node structure

Case 1: A activates B (by unspecified mechanism)

Standard notation

Reduced notation
Species type, Reaction type is stored in `<annotation>` for each species, reactions

Layout information is stored separately

```xml
<sbml>
  <model>
    <annotation>
      layout information
    </annotation>
    <listOfSpecies>
      <species>
        <annotation>species type</annotation>
      </species>
    </listOfSpecies>
  </model>
</sbml>
```
Graphical Notation ↔ SBML
SBML w/ or w/o Graphical Notation

Pure SBML (w/o SBGN)

w/ SBGN
CellDesigner 3.5.1

- SBML support
- Graphical notation (SBGN)
- Built-in simulator (SBML ODE Solver)
- Integrate with Analysis tool, other simulators through SBW
- Database connection
- Export to PDF, PNG, etc.
- Freely available

Supported Environment
- Windows (2000 or later)
- Mac OS X
- Linux

[http://celldesigner.org](http://celldesigner.org)
CellDesigner 4.0 beta

- Enhanced graphical notation (SBGN Level-1 draft)
- Layer function
- Plugin development framework
- GUI improvement
Layer function

Add graphical / text object to your model
Enhanced Graphical Notation

CellDesigner 4.0 supports SBGN Level-1 proposal
GUI improvement

Enhanced Kinetic Law Editor

\[ v = \frac{V_m S}{K_m + S} \]
GUI improvement

Enhanced Kinetic Law Editor
GUI improvement

Macro function
Plugin development

- Develop plugin on Eclipse
- Call plugin from [Plugin] menu on CellDesigner
Please download CellDesigner 4.0 alpha from

http://celldesigner.org/
Installation
Create new model:

- [File] → [New] → input title → [OK]
Enable [Grid Snap] will help you draw your model much easier.
Create Reaction

Create Protein “A” and “B”

Draw “State transition” arrow from “A” to “B”
Add Anchor Point

- Add 2 anchor points to reaction
- Drag reaction and anchor point to change its shape
Add Catalysis reaction

- Add Protein “C”
- Add Catalysis reaction from “C” to the reaction
Set Active state

Select Protein “B”

[Component] → [Set Active]
Right click on Protein “C”
Select [Change Color & Shape...]
Compartment

- Click [Compartment] icon
- Drag mouse cursor to specify its area
- Input name of compartment
Add Residue to Protein

- Create new model (test2)
- Create Protein “A”
- Select Protein “A” in [Proteins] Tab
- Click [Edit] button
Add Residue to Protein

- Click [add] button on [Protein] dialog
- Input name for the residue (tst1)
- Click [Close] button
- Click [Update] Button
Add Residue to Protein

- Copy & Paste Protein “A” and then draw “State Transition” arrow
- Right Click on “A” (right side) and select [Change Identity...]
- Click residue “tst1” in Dialog
- Select [phosphorylated] in modification
Change position of Residue

- Select Protein “A” in [Proteins] Tab
- Click [Edit] button
- Click residue “tst1” in Dialog
- Click [edit] button
- Drag [angle] slidebar
Create new model (test3)
Create Proteins “A” and “B”
Copy & Paste both “A” and “B”
Click [Complex] icon and create complex “C”

Drag Protein “A” and “B” into complex C

Draw “Association” arrow
Create new model (test4)
Create gene, RNA and Protein
Draw “Transcription” and “Translation”

See “geneRNA.xml” for more examples
Database connection

Search Database by Name:

- SGD
- DBGET
- iHOP
- Entrez Gene
- Genome Network Platform

Diagram:

- YHR023W
- caf1
- 2.1.3.2
- C00049
- L-glutamate
- cdc28
Search Database by Notes:

- PubMed: PMID: 123456
- Entrez Gene: GeneID: 4015
Database connection

Import model from BioModels.net
Auto layout

[File] → [Open] → samples/MAPK.xml

[Layout] → [Orthogonal Layout]
Create following biochemical reaction

Click [Simulation] → [ControlPanel] and call SBML ODE Solver

\[ \frac{d[B]}{d[t]} = k \cdot [A] \]

\[ k = 0.3 \]
\[ A = 0.1 \]
\[ B = 0 \]
Create new model (ex1)
Create reaction
Right click on the reaction and select [Edit KineticLaw...]
Click [New] button on [Parameters] tab

Input values as follows:

- id: k
- name: k
- value: 0.3

\[\frac{d[B]}{dt} = k \times [A]\]

k = 0.3
A = 0.1
B = 0
Select parameter “k”
Click top most text field
Click [copy] button
Click [*] button
Select Protein “A”
Click top most text field
Click [copy] button

\[ \frac{d[B]}{d[t]} = k \times [A] \]

\[ k = 0.3 \]
\[ A = 0.1 \]
\[ B = 0 \]
Double click [initialQuantity] column for Protein “A”

Set value as 0.1

\[ \frac{d[B]}{dt} = k \times [A] \]

\[ k = 0.3 \]
\[ A = 0.1 \]
\[ B = 0 \]
Click [Simulation] → [ControlPanel]
Set [End Time] to 20
Click [Execute] button
Create following biochemical reactions

Execute simulation from [ControlPanel]
Simulation (ex2)

Change parameter k1 to 30.0

- **A**: 0.5
- **B**: 0.2
- **C**: 0.01
- **D**: 0.02
- **E**: 0
- **F**: 0

- **k1**: 30.0
- **k2**: 0.01
- **k3**: 0.6

0 < t < 100

---

Graphs comparing the change in parameter k1 from 0.3 to 30.0 on the amount of substances over time.
Simulation (ex2)

- Click [Parameters] tab
- Double click [Value] column for k1
- Change parameter k1 to 30.0
Click [Interactive Simulation] tab
Click [Parameter value] radio button
Click [Define Range] button
Click [Max] column for k1 and set value as 3.0
Drag sliderbar for k1
Plugin development
Plugin development

- Develop plugin on Eclipse
- Call plugin from [Plugin] menu on CellDesigner
CellDesigner

- Add / modify object (species, reaction, etc.)

Get object (species, reaction, etc.) information

Plugin
Development environment

- CellDesigner 4.0 alpha
- JDK 1.5.0 or 1.4.2 (for MacOSX 10.3)
- Eclipse (tested on 3.2.1)
How to Install Plugins

- Copy plugin file (.jar file) to CellDesigner’s plugin folder
  - Windows: C:/Program Files/CellDesigner4.0alpha/plugin
  - MacOSX: /Applications/CellDesigner4.0alpha/plugin
Sample plugin

- Copy `sample_plugin.jar` in `samples/plugin/jar` folder to `plugin` folder
- Restart CellDesigner
Sample plugin

- [File] → [Open] → samples/MAPK.xml
- [Plugin] → [Sample Plugin1] → [Open Sample Plugin1 dialog]
- Select **MKKK** and click [GET]
Sample plugin

- Create new model
- Input Species Information and click [ADD]
How to build your plugin

- Download Eclipse 3.2.1 from http://www.eclipse.org/
- Launch Eclipse and specify your workspace (ex. Desktop/workspace)
- Click [Workbench] icon
Create new project

1. [File] → [New] → [Project]
2. Select “Java Project” and click [Next]
3. Input “Project name” (MyPlugin) and select [Create separate source and output folders]
Import sample source

- Click [+ ] button next to [MyPlugin]
- Right click “src” folder and click [Import]
- Select [File system] and click [Next]
Import source file

1. Click [Browse] button next to “From directory”
2. Select “C:\Program Files\CellDesigner4.0alpha\samples\plugin\src” and click [OK]
3. Click check box next to “src” folder
Select Java Build Path

- Right click [MyPlugin] → [Properties]
- Click [Java Build Path] and click [Libraries] tab
- Click [Add External JARs] button
Select Java Build Path

Select following .jar files

- C:\Program Files\CellDesigner4.0alpha\exec\celldesigner.jar
- C:\Program Files\CellDesigner4.0alpha\lib\sbmlj.jar
Imported java source files are automatically compiled and java class files are generated in the "bin" directory of your project directory

NG

OK
Generate jar files

- Right click [MyPlugin] → [Export]
- Select [JAR file] and click [Next]
Generate jar files

- Check your project (MyPlugin)
- Select [Export generated class files and resources]
- Specify JAR file

Put jar file to plugin folder
How to implement plugin

- Write your plugin class
  - extend `CellDesignerPlugin` class

- Write an action class
  - extend `PluginAction` class

- Create menu and menu item
  - use `PluginMenu`, `PluginMenuItem`

- Register PluginMenu to CellDesigner
  - use `addCellDesignerPluginMenu()`

- Implement some methods to receive events from CellDesigner
1. Write your plugin class

Your plugin class must extend the `CellDesignerPlugin` class. CellDesigner will call the constructor of your plugin class to instantiate it.

```java
public class SamplePlugin extends CellDesignerPlugin {
    // Constructor
    public SamplePlugin() {
    }
}
```
2. Write action class

Write an action class which extends the `PluginAction` class for an action event that would be passed when the plugin menu is selected on CellDesigner.

```java
public class SampleAction extends PluginAction {

    public SampleAction(SamplePlugin plugin) {
        // Write your code for constructor
    }

    public void myActionPerformed(ActionEvent e) {
        // Write your code for action event
    }
}
```
3. Create menu and item

Use `PluginMenu` class and `PluginMenuItem` class to create menus on CellDesigner. Register the action class to the `PluginMenuItem` for CellDesigner to invoke the action.

```java
public class SamplePlugin extends CellDesignerPlugin {
    // Constructor
    public SamplePlugin() {
        PluginMenu menu = new PluginMenu("Sample");
        SampleAction action = new SampleAction(this);
        PluginMenuItem item = new PluginMenuItem("Sample1", action);
        menu.add(item);
        addCellDesignerPluginMenu(menu);
    }
}
```
4. Register PluginMenu

Use following methods to register PluginMenu to CellDesigner

- `addCellDesignerPluginMenu()`
  - Register menu to Plugin menu
- `addSpeciesPopupMenu()`
- `addReactionPopupMenu()`
- `addCompartmentPopupMenu()`
  - Register menu to right-clicked pop-up menu
5. Implement methods

Implement **following methods** to receive events from CellDesigner (**required**).

```java
public class SamplePlugin extends CellDesignerPlugin {
    public SamplePlugin() {}          // Constructor
    public void addPluginMenu() {}    // add PluginMenu

    public void SBaseAdded(PluginSBase sbase) {} 
    public void SBaseChanged(PluginSBase sbase) {} 
    public void SBaseDeleted(PluginSBase sbase) {} 
    public void modelOpened(PluginSBase sbase) {} 
    public void modelSelectChanged(PluginSBase sbase) {} 
    public void modelClosed(PluginSBase sbase) {} 
}
```
Plugin can get following information:

- Selected model (SBML)
  - `PluginModel getSelectedModel()`
- All opened model (SBML)
  - `PluginListOf getAllModels()`
- Selected node on model
  - `PluginListOf getAllSelectedAllNode()`
- All nodes on model
  - `PluginListOf getAllSpeciesNodes()`
You can implement functions to add, update and delete PluginSBase in CellDesignerPlugin. The Plugin can notify CellDesigner these changes via CellDesignerPlugin interface.

- `notifySBaseAdded(PluginSBase sbase)`
- `notifySBaseChanged(PluginSBase sbase)`
- `notifySBaseDeleted(PluginSBase sbase)`
Some actions trigger sequential actions. You have to implement the sequential actions in your plugin.

Example: delete species S2
private void getSelectedSpecies() {
    PluginListOf lof = plugin.getSelectedSpeciesNode();
    if (lof != null) {
        // get PluginSpeciesAlias
        PluginSpeciesAlias alias = (PluginSpeciesAlias)lof.get(0);

        // get position
        double x = alias.getX();
        double y = alias.getY();

        // get Species
        PluginSpecies sp = alias.getSpecies();
        String name = sp.getName();
        String id = sp.getId();
    }
}
Example plugin

http://celldesigner.org/~funa/plugintutorial.jar

- Get SpeciesAlias info
- Print out SpeciesAlias info (for debug)
- Change SpeciesAlias property
- Change color, size, position of Proteins depend on its name (work with MAPK.xml)
- Visualize InitialAmount
- Change color of Species when its InitialAmount < 20.0
Acknowledgment

SBML
* SBML community
* US (Caltech)
  Michael Hucka
  Ben Bornstein
  Bruce Shapiro
  Ben Kovitz
* UK (Univ. Hertfordshire)
  Andrew Finney
  Sarah Keating
* Japan
  Akiya Jouraku (Keio Univ.)

SBGN
* SBGN community
* Nicolas Le Novere (EBI)
* Michael Hucka (Caltech)
* Hiroaki Kitano (SBI)
* Yukiko Matsuoka (SBI)

CellDesigner
* SBI
  Yukiko Matsuoka
  Hiroaki Kitano
* Keio Univ.
  Akiya Jouraku
* MKI
  Norihiro Kikuchi
  Yusuke Tamaru

SBML ODE Solver (Univ. Vienna)
  Rainer Machne
  Christoph Flamm

SBW (Univ. Washington)
  Frank Bergmann
  Herbert Sauro