CellDesigner Tutorial

Akira Funahashi
Keio University
The Systems Biology Institute
22nd Aug. 2008
Overview

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

- How to build a model with CellDesigner
- How to create CellDesigner plugin
Model representation

Standard representation method of biological models

Database

Software tools

CellDesigner

Translator

RoadRunner

AutoLayout
CellDesigner

= CellDesigner
CellDesigner

Modeling tool for biochemical and gene-regulatory network
CellDesigner

Modeling tool for biochemical and gene-regulatory network
CellDesigner

= 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

Modifiers: M
Reactants: R
Products: P

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

Biochemical reaction

SBML
What does SBML look like?

Biochemical reaction

S1
What does SBML look like?

Biochemical reaction

S1

S2
What does SBML look like?

Biochemical reaction

```
<listOfSpecies>
  <species id="s1" name="s1" compartment="default"
    initialAmount="0" charge="0"/>
  <species id="s2" name="s2" compartment="default"
    initialAmount="0" charge="0"/>
</listOfSpecies>
```
What does SBML look like?

Biochemical reaction

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

What does SBML look like?

"k \times [S1]"

<listOfSpecies>
  <species id="s1" name="s1" compartment="default" initialAmount="0" charge="0"/>
  <species id="s2" name="s2" compartment="default" initialAmount="0" charge="0"/>
</listOfSpecies>
What does SBML look like?

```xml
<listOfReactions>
  <reaction id="re1" reversible="false" fast="false">
    <listOfReactants>
      <speciesReference species="s1"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="s2"/>
    </listOfProducts>
    <kineticLaw formula="k*s1"/>
  </reaction>
</listOfReactions>

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

Biochemical reaction

S1 \xrightarrow{k \times [S1]} S2
```
Over 130 software packages support SBML

https://sbml.org
SBGN

A Visual Notation for Network Diagrams in Biology

Representation of Biochemical and Cellular Processes studied in Systems Biology

http://sbgn.org
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...
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

```xml
<celldesigner:speciesAlias compartmentAlias="ca3" id="a1" species="s1">
  <celldesigner:activity>active</celldesigner:activity>
  <celldesigner:bounds h="40.0" w="80.0" x="559.0" y="184.0">
  </celldesigner:bounds>
  <celldesigner:singleLine width="1.0"/>
  <celldesigner:paint color="#ffbb3d2ff" scheme="Gradation"/>
</celldesigner:speciesAlias>
```

(559.0, 84.0)

S1

h="40.0"

w="80.0"
SBML w/ or w/o Graphical Notation

Pure SBML (w/o Graphical Notation)

w/ Graphical Notation
CellDesigner 4.0.1

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

Supported Environment
- Windows (XP or later)
- Mac OS X (Tiger, Leopard)
- Linux

http://cellDesigner.org
What’s new

- Enhanced graphical notation (SBGN Level-1 draft)
- Integration with COPASI
- Plugin development framework
- GUI improvement
- Layer function
- libSBML 3
Integration with COPASI

Can call COPASI as a solver
Integration with COPASCI

Can call COPASCI as a solver
Add graphical / text object to your model
Layer function

Add graphical / text object to your model
GUI improvement

Enhanced Kinetic Law Editor

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

Enhanced Kinetic Law Editor

```
math
k3 * MKK_P * MKK / (KK3 + MKK)
```

View mode

```
k3 * MKK_P * MKK
-----
KK3 + MKK
```

Copy + - * / ()
Plugin development

- Develop plugin on Eclipse
- Call plugin from [Plugin] menu on CellDesigner
Please download CellDesigner 4.0.1 from

http://celldesigner.org/
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
Select Protein “A” in [Proteins] Tab
Click [Edit] button
Click residue “tst1” in Dialog
Click [edit] button
Drag [angle] slidebar
Complex

- 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 “Heterodimer Association” arrow
Gene & RNA

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

See “geneRNA40.xml” for more examples
Database connection

Search Database by Name:

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

Search Database by Notes:

- PubMed: PMID: 123456
- Entrez Gene: GeneID: 4015
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]
**Simulation (ex1)**

Create following biochemical reaction

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

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

- \( k = 0.3 \)
- \( A = 0.1 \)
- \( B = 0 \)
Create new model (ex1)

Create reaction

Right click on the reaction and select [Edit KineticLaw...]
Simulation (ex1)

- Click [New] button on [Parameters] tab

- Input values as follows:
  - id: k
  - name: k
  - value: 0.3

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

\[
\begin{align*}
  k &= 0.3 \\
  A &= 0.1 \\
  B &= 0
\end{align*}
\]
Select parameter “k”
Click top most text field
Click [copy] button
Click [ ] button
Select Protein “A”
Click top most text field
Click [copy] button

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

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

Set value as 0.1

\[ \frac{dB}{dt} = k \times [A] \]

\[ k = 0.3 \]
\[ A = 0.1 \]
\[ B = 0 \]
Click [Simulation] → [ControlPanel]

Set [End Time] to 20

Click [Execute] button
Create the following biochemical reactions:

- $A = 0.5$
- $B = 0.2$
- $k_1 \cdot A \cdot B$
- $k_1 = 0.3$
- $C = 0.01$
- $D = 0.02$
- $k_2 \cdot C$
- $k_2 = 0.01$
- $k_3 \cdot D$
- $k_3 = 0.6$
- $E = 0$
- $F = 0$

Execute simulation from [ControlPanel] where $0 < t < 100$.  

Graphically, the reaction network is as follows:

- $A$ (source) connects to $B$ with reaction $k_1 \cdot A \cdot B$.
- $B$ connects to $C$ with reaction $k_1 = 0.3$.
- $C$ connects to $D$ with reaction $k_2 = 0.01$.
- $D$ connects to $F$ with reaction $k_3 = 0.6$.
- $E$ and $F$ are sink nodes.
Change parameter $k_1$ to 30.0

$A = 0.5$
$B = 0.2$

$k_1 = 0.3$
$k_1 = 30.0$

$k_2 = 0.01$
$k_3 = 0.6$

$C = 0.01$
$D = 0.02$

$E = 0$
$F = 0$

$0 < t < 100$

Graphical representation:

- A: $k_1 \cdot A \cdot B$
- B: $k_1 = 30.0$
- C: $k_2 \cdot C$
- D: $k_3 \cdot D$
- E: $E = 0$
- F: $F = 0$

Experiments with different values of $k_1$:

- $k_1 = 0.3$
- $k_1 = 30.0$
Simulation (ex2)

- Click [Parameters] tab
- Double click [Value] column for k1
- Change parameter k1 to 30.0
Simulation (ex2)

- 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 slider bar 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 or higher
- JDK 1.5.0 or higher
- Eclipse 3.4.0 (may work on earlier version)
How to Install Plugins

- Copy plugin file (.jar file) to CellDesigner’s plugin folder
  - Windows: C:/Program Files/CellDesigner4.0.1/plugin
  - MacOSX: /Applications/CellDesigner4.0.1/plugin
Sample plugin

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

1. [File] → [Open] → samples/MAPK.xml
2. [Plugin] → [Sample Plugin1] → [Open Sample Plugin1 dialog]
3. Select MKKK and click [GET]
Create new model

Input Species Information and click [ADD]
How to build your plugin

1. Download Eclipse 3.4 from http://www.eclipse.org/
2. Launch Eclipse and specify your workspace (ex. Desktop/workspace)
3. Click [Workbench] icon
Create new project

- [File] → [New] → [Project]
- Select “Java Project” and click [Next]
- 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

- Click [Browse] button next to “From directory”
- Select “C:\Program Files\CellDesigner4.0.1\samples\plugin\src” and click [OK]
- 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\CellDesigner4.0.1\exec\celldesigner.jar
- C:\Program Files\CellDesigner4.0.1\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
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

1. Write your plugin class
   - extend `CellDesignerPlugin` class

2. Write an action class
   - extend `PluginAction` class

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

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

5. 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) {}  
}
```
Accessible information

- Selected model (SBML)
  - `PluginModel` `getSelectedModel()`
- All opened model (SBML)
  - `PluginListOf` `getAllModels()`
- Selected node on model
  - `PluginListOf` `getSelectedAllNode()`
- All nodes on model
  - `PluginListOf` `getAllSpeciesNodes()`
Notification from Plugin

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
```java
private void getSelectedSpecies() {
    PluginListOf lof = plugin.getSelectedSpeciesNode();
    if (lof.size() != 0) {
        // 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
* Caltech
  Michael Hucka
  Ben Bornstein
  Bruce Shapiro
  Sarah Keating
* Keio Univ.
  Akiya Jouraku

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

SBML ODE Solver (Univ. of Vienna)
  Rainer Machine
  Christoph Flamm

SBW (Univ. of Washington)
  Frank Bergmann
  Herbert Sauro

COPASI (Univ. of Heidelberg)
  Ralph Gauges
  Sven Sahle
  Ursula Kummer