Module description

Plugins

CellType Plugin

This plugin is responsible for defining cell types and storing cell type information. It is a basic plugin used by virtually every CompuCell simulation. The syntax is straight forward as can be seen in the example below:

```
<Plugin Name="CellType">
    <CellType TypeName="Medium" TypeId="0"/>
    <CellType TypeName="Fluid" TypeId="1"/>
    <CellType TypeName="Wall" TypeId="2" Freeze=""/>
 </Plugin>
```

Here we have defined three cell types that will be present in the simulation: Medium,Fluid, Wall. Notice that we assign a number – TypeId – to every cell type. It is strongly recommended that TypeId are consecutive positive integers (e.g. 0,1,2,3...). Medium is traditionally given TypeId=0 but this is not a requirement.

Notice that in the example above cell type “Wall” has extra attribute **Freeze=””**. This attribute tells CompuCell that cells of “frozen” type will not be altered by spin flips. Freezing certain cell types is a very useful technique in constructing different geometries for simulations or for restricting ways in which cells can move. In the example below we have frozen cell types wall to create tube geometry for fluid flow studies.

VolumeFlex Plugin

Volume Flex plugin is more sophisticated version of **Volume Plugin**. While Volume Plugin treats all cell types the same i.e. they all have the same target volume and lambda coefficient, VolumeFlex plugin allows you to assign different lambda and different target volume to different cell types. The syntax for this plugin is straightforward and essentially mimics the example below.

```
<Plugin Name="VolumeFlex">
    <VolumeEnergyParameters CellType="Prestalk" TargetVolume="68" LambdaVolume="15"/>
    <VolumeEnergyParameters CellType="Prespore" TargetVolume="69" LambdaVolume="12"/>
    <VolumeEnergyParameters CellType="Autocycling" TargetVolume="80" LambdaVolume="10"/>
    <VolumeEnergyParameters CellType="Ground" TargetVolume="0" LambdaVolume="0"/>
    <VolumeEnergyParameters CellType="Wall" TargetVolume="0" LambdaVolume="0"/>
 </Plugin>
```

Notice that in the example above cell types “Wall” and “Ground” have target volume and coefficient lambda set to 0 – very unusual. That's because in this particular those cells are were frozen so the parameters specified for these cells do not matter. in fact it is safe to remove specifications for these cell types, but just for the illustration purposes we left

them.

Using VolumeFlex Plugin you can effectively freeze certain cell types. All you need to do is to put very high lambda coefficient for the cell type you wish to freeze. You have to be careful though , because if initial volume of the cell of a given type is different from target volume for this cell type the cells will either shrink or expand to match target volume (this is out of control and you should avoid it), and only after this initial volume adjustment will they remain frozen . That is provided lambdaVolume is high enough. in any case, we do not recommend this way of freezing cells because it is difficult to use, and also not efficient in terms of speed of simulation run.

VolumeLocal Plugin

VolumeLocal Plugin is very similar to **Volume** Plugin. You specify both lambda coefficient and target volume, but as opposed to Volume Plugin the energy is calculated using target volume cell attribute. That is **each cell** (not just each cell type) has a target volume attribute. In the course of simulation you can change this target volume depending on e.g. concentration of FGF in the particular cell. This way you can specify which cells grow faster , which slower based on a state of the simulation. This plugin requires you to develop a module (plugin or steppable) which will alter target volume for each cell. That's why it may be difficult to use for non C++ savvy person. However, in the nearest future when scripting language is built into a CompuCell , use of this plugin will be greatly simplified.
Example syntax:

```
<Plugin Name="VolumeLocal">
       <TargetVolume>27</TargetVolume>
       <LambdaVolume>2.5</LambdaVolume>
 </Plugin>
```

SurfaceFlex Plugin

SurfaceFlex plugin is more sophisticated version of **Surface Plugin**. Everything that was said with respect to **VolumeFlex** plugin applies to **SurfaceFlex**. For syntax see example below:

```
 <Plugin Name="SurfaceFlex">
  <SurfaceEnergyParameters CellType="Prestalk" TargetSurface="90" LambdaSurface="0.15"/>
  <SurfaceEnergyParameters CellType="Prespore" TargetSurface="98" LambdaSurface="0.15"/>
  <SurfaceEnergyParameters CellType="Autocycling" TargetSurface="92" LambdaSurface="0.1"/>
  <SurfaceEnergyParameters CellType="Ground" TargetSurface="0" LambdaSurface="0"/>
  <SurfaceEnergyParameters CellType="Wall" TargetSurface="0" LambdaSurface="0"/>
 </Plugin>
```

SurfaceLocal Plugin

Analog of VolumeLocal plugin but for surface. Again it requires module development in C++ . see the description of VolumeLocal plugin for more details.

NeighborTracker Plugin

This plugin , as its name suggests , tracks neighbors of every cell. In addition it calculates common contact area between cell and its neighbors. We consider a neighbor this cell that has at least one common pixel side with a given cell. This means that cells that touch each other either “by edge” or by “corner” are not considered neighbors. See the drawing below:

This plugin is used as a helper module by other plugins and modules e.g. Viscosity and AdvectionDiffusionSolver use NeighborTracker plugin.

CellVelocity Plugin

A major function for this plugin is to attach new attribute (CellVelocityData) to every cell. CellVelocityData attribute is an object that stores history of cell's center of mass , and a velocity of the cell.
Example usage is shown below:

```
<Plugin Name="CellVelocity">
   <VelocityDataHistorySize>2</VelocityDataHistorySize>
   <EnoughDataThreshold>2</EnoughDataThreshold>
</Plugin>
```

`VelocityDataHistorySize` specifies how many centers of mass are to be stored as a history of cell's center of mass. For simple average velocity extraction 2 is sufficient. Note that the bigger the history size the more memory simulation will require. This could be an important factor on older machines or machines with small RAM memory.

EnoughDataThreshold tells CompuCell or CompuCellPlayer how many times cell's center of mass has o be updated in order to be able to reliably estimate cell's velocity. This parameter has to be greater or equal 2 and 2 is sufficient in most cases. There is no memory penalty in specifying greater that 2 EnoughDataThreshold , you will just have to wait longer to get velocity estimate. Sometimes such a behavior maybe desirable.

In CompuCell CellVelocity plugin is used in conjunction with CellVelocity Steppable to extract average cell velocity. See example below:

```
.
.
.

<Plugin Name="CellVelocity">
    <VelocityDataHistorySize>2</VelocityDataHistorySize>
    <EnoughDataThreshold>2</EnoughDataThreshold>
 </Plugin>
.
.
.
 <Steppable Type="CellVelocity">
    <UpdateFrequency>50</UpdateFrequency>
 </Steppable>
```

The way it works is the following:
Every predefined number of Monte Carlo steps (MCS) (specified by UpdateFrequency - here it is 50) CellVelocity Steppable examines each cell and calculatesa displacement of the center of mass. Next it divides this displacement by the time , measured in MCS to get the velocity. This velocity is stored inside CellVelocityData cell attribute. Once you use CellVelocity Plugin and CellVelocity Steppable you will be able to visualize velocity field in the Player.

CAUTION: One has to make sure that UpdateFrequency is not too big. By that we mean that on average cell should not move more than one lattice length during each update period (given by UpdateFrequency). If the cells moves more than that (e.g. fluid flow in the pipe with periodic boundary conditions along 'x' coordinate) then the value of average velocity will be corrupted.

CellInstantVelocity Plugin.

This plugin similarly as CellVelocity plugin attaches new attribute (CellVelocityData) to every cell. However, in addition to this, every spin flip (not MCS , MCS consists of many spin flips) updates center of mass history and calculates instantenous velocity which in this case is equal to a difference between center of mass of a cell before and after the spin flip.

Chemotaxis

Chemotaxis plugin , as its name suggests is used to simulate chemotaxis of cells. For every spin flip this plugin calculates change of energy associated with pixel move. The energy contribution comes from cell whose pixel is assigned to new site. In CompuCell3D terminology it is newCell.
Energy is proportional to chemotaxis coefficient lambda and concentration difference in target pixel its neighbor pixel. A neighbor pixel in this case is a pixel whose spin is assigned to a target pixel. In many situations it is a nearest neighbor, but this is not a requirement – user may specify spin flips to take place between more distant neighbors.

Let's look at the syntax by studying the example usage of the Chemotaxis plugin:

```
<Plugin Name="Chemotaxis">
   <ChemicalField Source="FlexibleDiffusionSolverFE" Name="FGF" Lambda="200" Flex="">
        <ChemotaxisByType Type="Amoeba" Lambda="300"/>
        <ChemotaxisByType Type="Bacteria" Lambda="-200"/>
   </ChemicalField>
 </Plugin>
```

The body of the chemotaxis plugin description contains sections called `ChemicalField` .In this section you tell CompuCell3D which module contains chemical field that you wish to use for chemotaxis. In our case it is `FlexibleDiffusionSolverFE` . Next you need to specify the name of the field - `FGF` in our case. Strength of chemotaxis is given by Lambda attribute name. Flex flag is used to tell CompuCell3D that different cells may have different chemotacting properties. In our case Amoeba goes up FGF gradient with chemotaxis strength 300 and Bacteria moves down the gradient with Lambda=-300. When such flexibility is not required i.e. when you assume that chemotaxis is the same for all the cells you may use the following syntax:

```
<Plugin Name="Chemotaxis">
   <ChemicalField Source="FlexibleDiffusionSolverFE" Name="FGF" Lambda="200" />
 </Plugin>
```

It is important to realize that you may simulate chemotaxis with respect to many fields. For example you may use the following syntax;

```
<Plugin Name="Chemotaxis">
   <ChemicalField Source="FlexibleDiffusionSolverFE" Name="FGF" Lambda="200" Flex="">
        <ChemotaxisByType Type="Amoeba" Lambda="300"/>
        <ChemotaxisByType Type="Bacteria" Lambda="-200"/>
   </ChemicalField>
   <ChemicalField Source="FlexibleDiffusionSolverFE" Name="Fibronectin" Lambda="100" />
 </Plugin>
```

Here Bacteria and amoeba respond differently to FGF but their response to Fibroectin is described by the same energy term.

CAUTION: when you use chemotaxis plugin you have to make sure that fields that you refer to and module that contains this fields are declared in the xml file. Otherwise you will most likely cause either program crash (which is not as bad as it sounds) or unpredicted behavior (much worse scenario, although unlikely as we made sure that in the case of undefined symbols, the program exits)

ChemotaxisDicty plugin.

The priniciples behind this plugin are the same as in the case of Chemotaxis plugin. The only difference is that in ChemotaxisDicty plugin calculates chemotaxis energy only during the periods when cells are allowed to chemotact. Those periods are set individually for every cell by module (steppable in this case) called DictyCemotaxisSteppable. DictyCemotaxisSteppable and Chemotaxis are examples of

CompuCell extension modules and were developed to facilitate Dictyoestelium morphogenesis simulation. Once scripting language will be built into CompuCell a plugin like ChemotaxisDicty would be most likely be coded using scripting language and where one would call “regular” Chemotaxis plugin.

The syntax of ChemotaxisDicty plugin is the following:

```
 <Plugin Name="ChemotaxisDicty">
   <Lambda> 200 </Lambda>
   <ChemicalField Source="ReactionDiffusionSolverFE_SavHog">cAMP</ChemicalField>
 </Plugin>
```

The symbols appearing here are analogous to the ones appearing in the “regular” Chemotaxis . The only difference is that here the name of the field (cAMP) is listed as a value of an xml element i.e. between xml tags `<ChemicalField...>cAMP</ChemicalField>`

Concentration Plugin

The sole purpose for this plugin as of now is to attach additional attribute (of type float) to the cell object.
As such this plugin may be used by CompuCell3D developers, but is not directly usable from xml level. This of course will be changed once scripting language becomes available.

ContactFlex

The syntax of this plugin is the following:

```
 <Plugin Name="ContactFlex">
   <Energy Type1="Medium" Type2="Medium">0</Energy>
   <Energy Type1="Amoeba" Type2="Amoeba">15</Energy>
   <Energy Type1="Amoeba" Type2="Medium">8</Energy>
   <Energy Type1="Bacteria" Type2="Bacteria">15</Energy>
   <Energy Type1="Bacteria" Type2="Amoeba">15</Energy>
   <Energy Type1="Bacteria" Type2="Medium">8</Energy>
   <Depth>2</Depth>
   <Haptotaxis Source="FlexibleDiffusionSolverFE" Name="FGF"
HaptotactingTypes="Medium,Amoeba">1.0</Haptotaxis>
 </Plugin>
```

As one can see this is very much like usual contact plugin except that now we see new xml tagt there :

```
   <Haptotaxis Source="FlexibleDiffusionSolverFE" Name="FGF"
HaptotactingTypes="Medium,Amoeba">1.0</Haptotaxis>
```

The energy calculations that this plugin does are the same as in the case of Contact plugin . However there is additional term that modifies contact energy. This term depends on the the value of concentration of a chemical field (FGF in this example).

Put exact formula here.

HaptotaxisTypes types allows user to specify which cell types are subject to this extra contact energy modification.Names of the cell types are listed separated by “,” with no spaces.

ExternalPotential plugin

This plugin is responsible for imposing a directed pressure (or rather force) on cells. It is used mainly in fluid flow studies with periodic boundary conditions along these coordinates along which force acts. If NoFlux boundary conditions are set instead , the cells will be squeezed.

This is the example usage of this plugin:

```
<Plugin Name="ExternalPotential">
   <Lambda x="-0.5" y="0.0" z="0.0"/>
</Plugin>
```

Lambda is avector quantity and determines components of force along three axes. In this case we apply force along x axis and in this particular simulation in the <Potts> section of the simulation description we set boundary conditions along “x” axis.

```
<Potts>
        .
        .
        .
   <Boundary_x>Periodic</Boundary_x>
    <FlipNeighborMaxDistance>1</FlipNeighborMaxDistance>
 </Potts>
```

SimpleClock Plugin

The main purpose of this plugin is to attach additional attribute – clock – of type SimpleClock (essentially consists of integer counter and a flag) . It is used in all those simulation in which cells exhibit time dependent behavior. For example in the Dictyoestelium simulation cells chemotact for certain amount of monte Carlo steps, after the cAMP concentration reaches threshold level. In somitogenesis simulations there one may introduce cell clock which would guides segmentation process.

ViscosityPlugin

The detailed description of internals of this plugin can be found in the paper by Debasis Dan, James Glazier, Chris Mueller and Kun Chen: Phys. Rev. E 72, 041909 (2005) This is fairly complicated plugin and required developing of few modules (e.g. NeighborTracker). This plugin essentially simulates cells' viscosity. One can use flow2D.xml demo simulation to see how viscosity works in the case of fluid flow. Try running with viscosity and then comment out the viscosity declaration in the xml file to see the difference.

The syntax for this plugin is straightforward (as opposed to actual implementation):

```
<Plugin Name="Viscosity">
   <LambdaViscosity>10.0</LambdaViscosity>
</Plugin>
```

LambdaViscosity parameter determines how viscous cells are (the bigger the more viscous).

Steppables.

Steppables are CompuCell modules that are called every Monte Carlo Step (MCS). More precisely, they are called after all the spin attempts in a given MCS have been carried out. Steppables may have various functions like for example solving PDE's, checking if critical concentration threshold have been met, updating target volume or target surface given the concentration of come growth factor, initializing cell field, writing numerical results to a file etc. In summary Steppables perform all functions that need to be done every MCS. In the reminder of this section we will present steppables currently available in the CompuCell and describe their usage.

AdvectionDiffusionSolver steppable.

This steppable solves advection diffusion equation on a cell field as opposed to grid. Of course, the inaccuracies are bigger than in the case of PDE being solved on the grid but on the other hand solving the PDE on a cell field means that we associate cocentration with a given cell (not just with a lattice point). This means that as cells move so does the concentration. In other words we get advection for free. The mathematical treatment of this kind of approximation was spelled out in Phys. Rev. E 72, 041909 (2005) paper by D.Dan et al.

In addition to just solving advection-diffusion equation this module allowes users to specify secretion rates of the cells as well as different secretion modes. More about it in a moment. First let's see how one uses AdvectionDiffusionSolver:

This is an example syntax:

```
<Steppable Type="AdvectionDiffusionSolverFE">
     <DiffusionField>
        <DiffusionData>
            <FieldName>FGF</FieldName>
            <DiffusionConstant>0.05</DiffusionConstant>
            <DecayConstant>0.003</DecayConstant>
            <ConcentrationFileName>flowFieldConcentration2D.txt</ConcentrationFileName>
            <DoNotDiffuseTo>Wall</DoNotDiffuseTo>
        </DiffusionData>
        <SecretionData>
            <Secretion Type="Fluid">0.5</Secretion>
           <SecretionOnContact Type="Fluid"
SecreteOnContactWith="Wall">0.3</SecretionOnContact>
        </SecretionData>
```

```
    </DiffusionField>
 </Steppable>
```

Inside AdvectionDiffusionSolver you need to define sections that describe a field on which the steppable is to operate. In our case we declare just one diffusion field. Inside the diffusion field we specify sections describing diffusion and secretion. Let's take a look at DiffusionData section first

```
        <DiffusionData>
            <FieldName>FGF</FieldName>
            <DiffusionConstant>0.05</DiffusionConstant>
            <DecayConstant>0.003</DecayConstant>
            <ConcentrationFileName>flowFieldConcentration2D.txt</ConcentrationFileName>
            <DoNotDiffuseTo>Wall</DoNotDiffuseTo>
        </DiffusionData>
```

We give a name (FGF) to the diffusion field – this is required as we will refer to this field in other modules. Next we specify diffusion constant and decay constant.

CAUTION: We use Forward Euler Method to solve these equations. This is not a stable method for solving diffusion equation and we do not perform stability checks. If you enter too high diffusion constant for example you may end up with unstable (wrong) solution. Always test your parameters to make sure you are not in the unstable region.

`ConcentrationFileName` is an optional tag and lets you specify a text file that contains a values of concentration for every pixel. The value of concentratio of the last pixel read for a given cell becomes an overall value of concentration for a cell. That is if cell has , say 8 pixels and you specify different concentration at every pixel, then cell concentration will be the last one read from the file.

Concentration file format is the following:
x y z c

where x,y,z, denote coordinate of the pixel. c is the value of the concentration.

Example:

0 0 0 1.2
0 0 1 1.4
...

You may also specify cells which will not participate in the diffusion. You do it using `<DoNotDiffuseTo>` tag. In this example you do not let any FGF diffuse into Wall cells. You may of course use as many as necessary `<DoNotDiffuseTo>` tags .

In addition to diffusion parameters we may specify how secretion should proceed. SecretionData section contains all the necessary information to tell CompuCell how to handle secretion. Let's study the example:

```
<SecretionData>
  <Secretion Type="Fluid">0.5</Secretion>
  <SecretionOnContact Type="Fluid" SecreteOnContactWith="Wall">0.3</SecretionOnContact>
</SecretionData>
```

Here we have a definition two major secretion modes. Line `<Secretion Type="Fluid">0.5</Secretion>` ensures that every cell of type Fluid will get 0.5 increase in concentration every MCS. Line `<SecretionOnContact Type="Fluid" SecreteOnContactWith="Wall">0.3</SecretionOnContact>` means that cells of type Fluid will get additional 0.3 increase in concentration but only when they touch cell of type Wall. This mode of secretion is called SecretionOnContact.

AmoebaeFieldInitilizer steppable

This steppable is responsible for initializing cell field for amoeba simulation. It is an example steppable whose functionality can be entirely shifted to PIF initializer where one writes a script with cell field description. in the future this steppable will not be supported and instead better alternatives will be presented.
The syntax is straightforward:

```
<Steppable Type="AmoebaeInitializer">
   <Radius>1</Radius>
   <Position x="10" y="5" z="10"/>
</Steppable>
```

Radius denotes initial radius of the cell and Position tag defines initial center of the cell.

CellVelocity Steppable

This steppable is used to calculate average cell velocity. It uses CellVelocity plugin to get access to CellVelocityData cell attribute. The syntax is straightforward:

```
<Steppable Type="CellVelocity">
   <UpdateFrequency>10</UpdateFrequency>
</Steppable>
```

UpdateFrequency tag lels you define how often cell velocity is to be updated. As discussed in plugin section (see discussion on CellVelocity plugin) one has to make sure that update frequency is not too large as it may result in miscalculations. On the other hand too low update frequency may cause average velocity to look noisy. Again you need find optimal values by running test simulations.

DictyChemotaxis Steppable

This steppable is used for dictyoestelium morphogenesis simulation. Every MCS it

examines concentration of a chemical and if it is above threshold level . This plugin also decrements and reloads cell's clock. Clock is reloaded when it is zero and concentration of the chemical (here, cAMP) is above threshold (here, 0.2) and at this moment cell is marked for being able to chemotact. Now, cell marked for chemotaxis will chemotact for a certain number of MCS given by the difference of values of tags `ClockReloadValue` and `ChemotactUntil`.
Let's look at the example:

```
<Steppable Type="DictyChemotaxisSteppable">
   <ClockReloadValue>850</ClockReloadValue>
   <ChemotactUntil>750</ChemotactUntil>
   <IgnoreFirstSteps>500</IgnoreFirstSteps>
   <ChetmotaxisActivationThreshold>0.2</ChetmotaxisActivationThreshold>
   <ChemicalField Source="ReactionDiffusionSolverFE_SavHog">cAMP</ChemicalField>
</Steppable>
```

ClockReloadValue is a reload value for the clock after above-mentioned conditions have been met. When the clock falls below `ChemotactUntil` value, cell does not chemotact, it enters refrectory period.

`IgnoreFirstSteps` tag is used to deactivate this chemotaxis in all the cells for firstcertain number of initial MCS's (here, 500)

Chemical field based on which DictyChemotaxis steppable makes its decisions is supplied by the line:
`<ChemicalField Source="ReactionDiffusionSolverFE_SavHog">cAMP</ChemicalField>`
where Source attribute contains a name of the module that declares given chemical field – here it is a steppable called `ReactionDiffusionSolverFE_SavHog` and the name of the field (specified as a content of xmlChemcalField tag) is cAMP.

DictyFieldInitializer

This is used to initialize cell field for Dictyoestalium simulation. This is an example:

```
<Steppable Type="DictyInitializer">
  <Gap>1</Gap>
  <Width>4</Width>
  <ZonePoint x="14" y="14" z="3" >10</ZonePoint>
  <PresporeRatio>0.8</PresporeRatio>
</Steppable>
```

width and gap values determine length of the cube and space between the cubes that represent cells. ZonePoint defines approximately where autocycling cells are to be placed and value 10 determines how much space they will occupy (user should experiment with the values to make sure that initial placement of cells is the right one). Prespore ration determines a percentage of prespore cells.

FlexibleDiffusionSolver

This steppable is one of the basis and most important modules in CompuCell simulations.

As the name suggests it is responsible for solving diffusion equation but in addition to this it also handles chemical secretion which by itself maybe thought of as being part of general diffusion equation. The principles of operations are analogous as in the case of AdvectionDiffusionSolver so most of has been said there3 applies to FlexibleDiffusionSolve. Also syntax is very similar. Let's see an example

```
 <Steppable Type="FlexibleDiffusionSolverFE">
     <DiffusionField>
        <DiffusionData>
            <FieldName>FGF8</FieldName>
            <DiffusionConstant>0.1</DiffusionConstant>
            <DecayConstant>0.002</DecayConstant>
            <DeltaT>0.1</DeltaT>
            <DeltaX>1.0</DeltaX>
            <DoNotDiffuseTo>Bacteria</DoNotDiffuseTo>
        </DiffusionData>
        <SecretionData>
            <Secretion Type="Amoeba">0.1</Secretion>
        </SecretionData>
    </DiffusionField>

    <DiffusionField>
        <DiffusionData>
            <FieldName>FGF</FieldName>
            <DiffusionConstant>0.02</DiffusionConstant>
            <DecayConstant>0.001</DecayConstant>
            <DeltaT>0.01</DeltaT>
            <DeltaX>0.1</DeltaX>
        <DoNotDiffuseTo>Bacteria</DoNotDiffuseTo>
        </DiffusionData>
        <SecretionData>
            <SecretionOnContact Type="Medium"
SecreteOnContactWith="Amoeba">0.1</SecretionOnContact>
            <Secretion Type="Amoeba">0.1</Secretion>
        </SecretionData>
    </DiffusionField>
 </Steppable>
```

We can see new xml tags `<DeltaT>` and `<DeltaX>`. Their values determine the correspondence between MCS and actual time and between lattice spacing and actual spacing size. In this example for the first diffusion field one MCS corresponds to 0.1 units of actual time and lattice spacing is equal 1 unit of actual length. What is happening here is that the diffusion constant gets multiplied by `DeltaT/ (DeltaX* DeltaX)`provided the decay constant is set to 0. If the decay constant is not zero `DeltaT` appears additionally in the term containing decay constant so in this case it is more than simple diffusion constant rescaling.

SecretionData sections are analogous to those defined in AdvectionDiffusionSolver. here however, the secretion is done done on per-pixel basis (as opposed to per cell basis for AdvectionDiffusionSolver). For example when we use the following xml statement
`<Secretion Type="Amoeba">0.1</Secretion>`
this means that every pixel that belongs to cells of type Amoebae will get boost in concentration by 0.1. That is the secretion proceeds uniformly in the whole body of a cell.

Alternative secretion mode would be the one described by the following line:
`<SecretionOnContact Type="Medium" SecreteOnContactWith="Amoeba">0.1</SecretionOnContact>`

Here the secretion will take place in medium and only in those pixels belonging to Medium that touch directly Amoeba.

More secretion schemes will be added in the future.

ReactionDiffusionSolver_SavHog

This is a steppable that solves reaction diffusion set of PDE's as given in the paper by P.Hogeweg and N.Savill **Modelling morphogenesis: from single cells to crawling slugs. J. theor. Biol. 184, 229-235.**
This steppable is used in Dictyostaelium simulation based on above – mentioned paper. Let's take a look at the syntax:

```
<Steppable Type="ReactionDiffusionSolverFE_SavHog">
    <NumberOfFields>3</NumberOfFields>
    <FieldName>cAMP</FieldName>
    <FieldName>Refractoriness</FieldName>
    <DeltaX>0.37</DeltaX>
    <DeltaT>0.01</DeltaT>
    <DiffusionConstant>1.0</DiffusionConstant>
    <DecayConstant>0.0</DecayConstant>
    <MaxDiffusionZ>8</MaxDiffusionZ>
    <IntervalParameters c1="0.0065" c2="0.841"/>
    <fFunctionParameters C1="20" C2="3" C3="15" a="0.15"/>
    <epsFunctionParameters eps1="0.5" eps2="0.0589" eps3="0.5"/>
    <RefractorinessParameters k="3.5" b="0.35"/>
    <MinDiffusionBoxCorner x="0" y="0" z="0"/>
    <MaxDiffusionBoxCorner x="40" y="40" z="8"/>
 </Steppable>
```

The syntax is a little bit different than in the case of diffusion solvers. You need to declare how many fields you will be using (two PDE give two fields, and we need also scratch field total of 3 fields is required). Next you specify PDE field names , do not give name to scratch field. subsequently you specify DeltaX and DeltaT and diffusion constant. As it was mentioned earlier Specification of DeltaX and DeltaT has an effect of rescaling the diffusion constant as decay constant is 0.
Line `<MaxDiffusionZ>8</MaxDiffusionZ>` requires some explanation. MaxDiffusionZ specifies region in which Refractoriness is allowed to diffuse. Refractoriness for pixels with z>=8 is set to 0. In this case this is set of xy planes for which z < 8. Similarly lines

```
    <MinDiffusionBoxCorner x="0" y="0" z="0"/>
    <MaxDiffusionBoxCorner x="40" y="40" z="8"/>
```

describe a 3D rectangular region in which the cAMP and refractoriness may diffuse. Parameters given in lines:

```
   <IntervalParameters c1="0.0065" c2="0.841"/>
    <fFunctionParameters C1="20" C2="3" C3="15" a="0.15"/>
    <epsFunctionParameters eps1="0.5" eps2="0.0589" eps3="0.5"/>
    <RefractorinessParameters k="3.5" b="0.35"/>
```

are described in the paper by Savill and Hogeweg.
In summary this steppable is a custom made module used in simulation of Dictyostelium

morphogenesis. It is not a general purpose so its use maybe limited. Availbility of scripting language would most likely lead to greater flexibility.