



Nancy Darling & Richard Salter

Oberlin College Oberlin, OH US

This work was supported by the National Science Foundation
(NSF Grant Number 0939153)

OMI
Oberlin Modeling Initiative

Nøva

Oberlin Modeling Initiative (OMnI)

- NSF funded program to introduce **dynamic systems thinking** across the Oberlin curriculum
 - *How do you think about complex problems where key elements interact and feed back into each other?*
- Challenge
 - Build a tool to support dynamic systems thinking for a broad, multi-disciplinary user group
 - Teach **thinking** not **programming**
 - Support collaborative, multi-disciplinary problem solving

Nova concept



- Flexible modeling platform suitable for student exercises but powerful enough for serious research applications
- Capable of a full range of dynamic models
 - Stock and flow
 - Spatial
 - Agent based
- Multi-platform
- Modular



No cost
license!

Nova: A Java-based platform that operates at multiple levels



Agent based
and spatial
modeling on a
stock & flow
core

Netlogo or
Agentsheets

View: Nova can operate entirely

Stella, Vensim,
Madonna

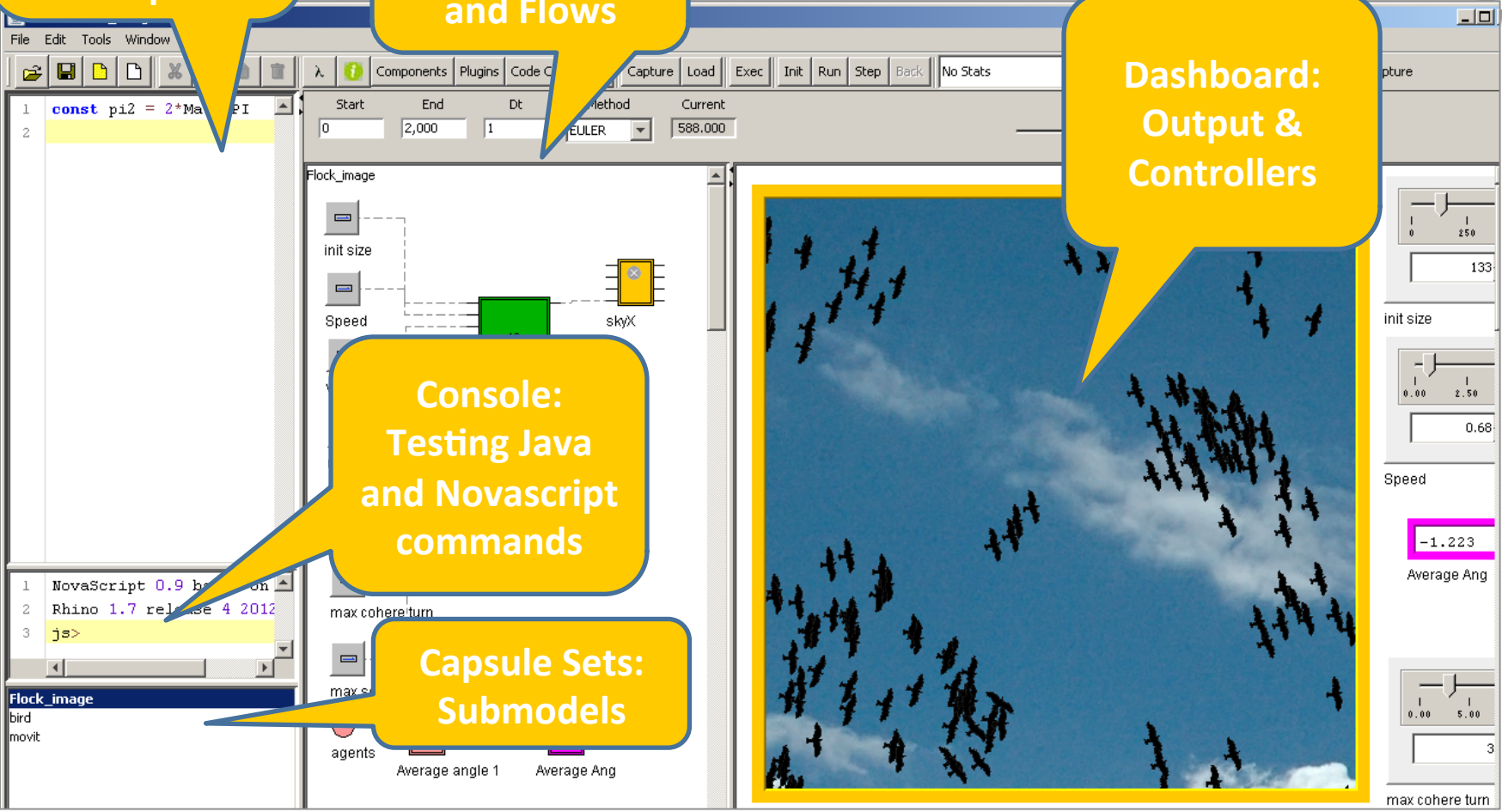
Programming
Window:
Novascript
functions &
scripts

Modeling
Canvas: Stocks
and Flows

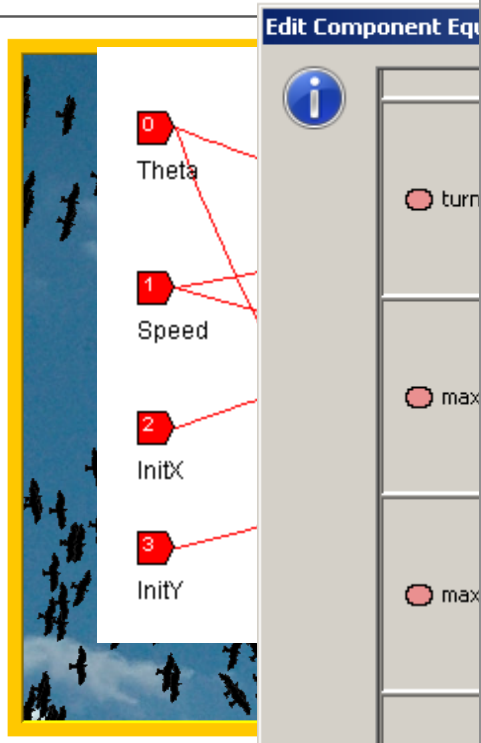
Dashboard:
Output &
Controllers

Console:
Testing Java
and Novascript
commands

Capsule Sets:
Submodels

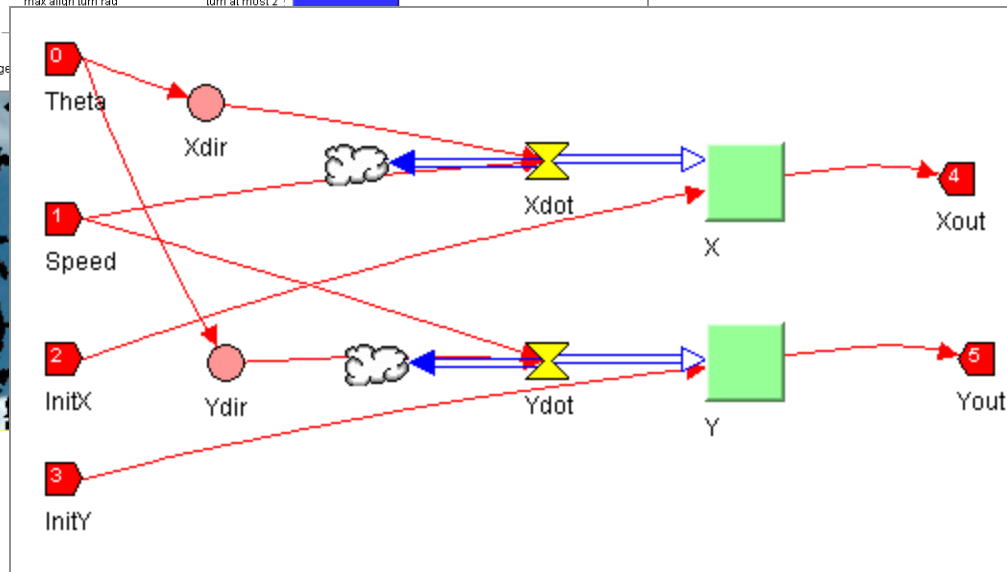
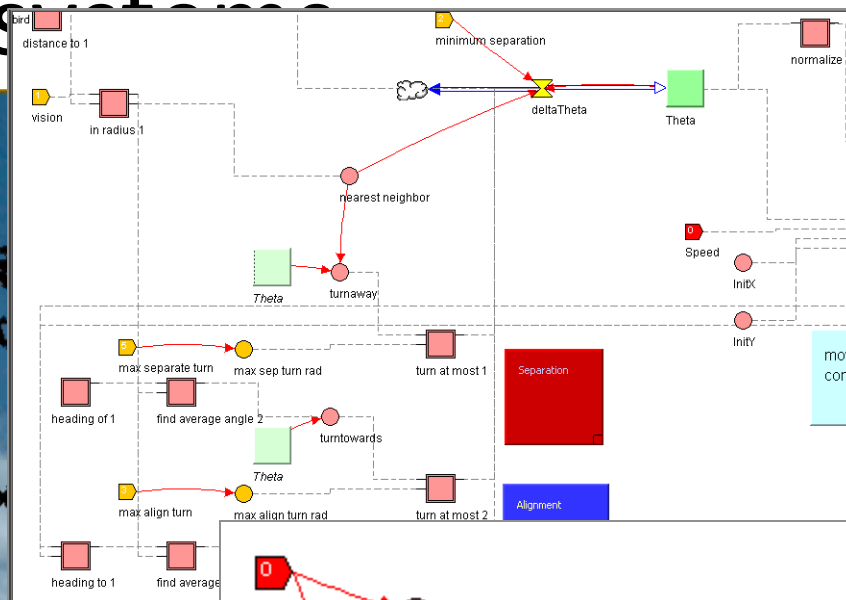
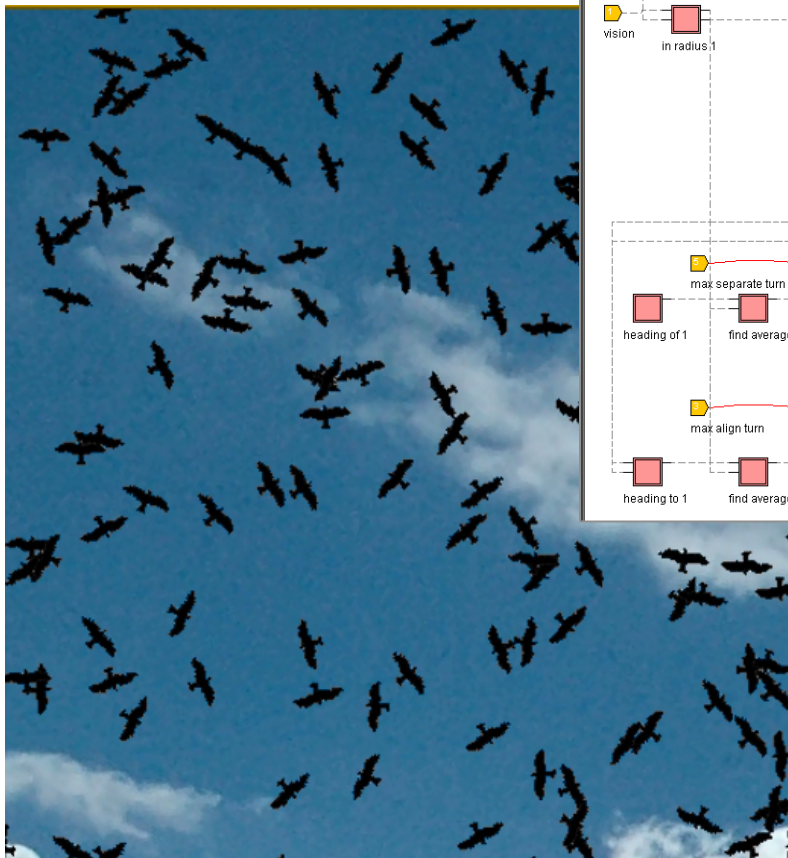


Overview: You can work with Nova at multiple levels



```
106 function(){
107     var f;
108     f = function(other) {
109         var theta = other.Theta;
110         var phi = theta % pi2;
111         if (phi > Math.PI) phi -= pi2;
112         else if (phi <= -Math.PI) phi += pi2;
113         return phi;
114     }
115     return {f:f};
116 },
117 ['f'], true, false),
118 in_radius: Dynamic(
119     function(rad,distanceTo){
120         var all,closest;
121         var best = Infinity;
122         var closest = 0;
123         var all = [];
124         for (var i = -rad; i < rad; i++)
125             for (var j = -rad; j < rad; j++) {
126                 var coords = CELL_COORDS();
127                 var y0 = coords.row + i;
128                 var x0 = coords.col + j;
129                 if (x0 < 0) x0 = cols + x0;
130                 if (y0 < 0) y0 = rows + y0;
131                 if (x0 >= cols) x0 = x0 - cols;
132                 if (y0 >= rows) y0 = y0 - rows;
133                 var agentset = AGENTS_AT(y0, x0);
134                 for (var k = 0; k < agentset.length; k++) {
135                     var z = agentset[k];
```

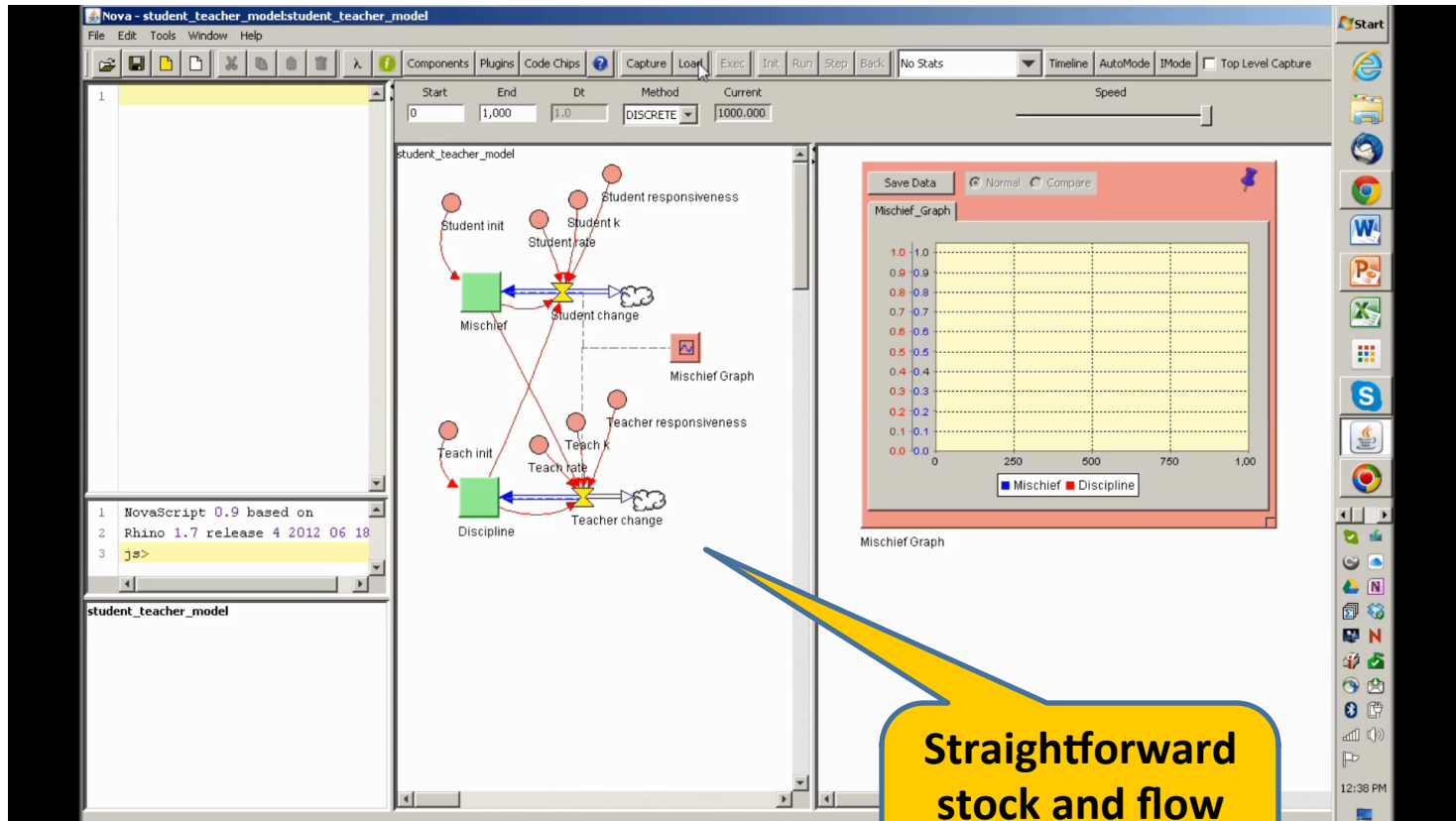
Overview: Capsules are what makes Nova interesting for people who work with nested systems



Advantages of Nova as a platform

- Allows people with different expertise to work at different components of the problem
- Allows stock & flow, spatial, and agent models simultaneously, allowing modeling of contagion
- Nested models through capsules and code chips, so you can move from the individual to group level and back down again
- Does not assume homogeneity of the population
- Automated runs across a range of distributions
- Output results graphically, csv, or directly into R
- Allows full integration of R and Java functions

Modularity in the Predator-Prey Model: **Mischiefous Students in a Classroom**



**Straightforward
 stock and flow
 model with
 output**

Working and Configured Controllers

The screenshot displays the Nova modeling environment. The main workspace shows a causal loop diagram for a student-teacher model. Key nodes include 'Student init', 'Student rate', 'Student k', 'Student responsiveness', 'Teacher init', 'Teacher rate', 'Teacher k', and 'Teacher responsiveness'. A central node labeled 'Mischief' is highlighted in green. A yellow starburst icon indicates a feedback loop. A graph window titled 'Mischief_Graph' is open, showing a plot with a y-axis from 0.0 to 1.0 and an x-axis from 0 to 1,000. The legend indicates 'Mischief' (blue) and 'Discipline' (red). Below the graph is a control panel with eight sliders and spinners for parameters: 'st ini' (0.19), 'st rate' (0.028), 'st resp' (0.1), 'st k' (1), 't ini' (0.1), 't rate' (-0.03), 't resp' (0.1), and 't k' (1).

After the model is developed, I can add easier to use sliders or spinners

the model

A simpler interface: capsules & chips

The image displays the Nova software interface, which is used for modeling and simulation. The interface is divided into several windows and panels:

- Top Panel:** Shows the simulation controls, including a menu bar (File, Edit, Tools, Window, Help), a toolbar with icons for capture, load, exec, init, run, step, back, and a speed slider. It also displays simulation parameters like Start (0), End (1,000), Dt (1), Method (EULER), and Current (1000.000).
- Left Panel:** Contains a component browser showing a hierarchy of objects: 'student_teacher' (containing 'st:ini' and 'st:ini:Student:Init') and 'Student_Teacher_Dyad' (containing 'student_teacher'). Below this is a code editor with NovaScript and Rhino 1.7 commands.
- Middle Panel:** Displays a diagram of the 'Student_Teacher_Dyad' capsule. It shows a central cyan capsule with two input ports labeled 'student rate' and 'teacher rate', and one output port labeled 'student teacher'. A 'Mischief Discipline' chip is connected to the output port.
- Right Panel:** Shows a graph titled 'Mischief Discipline' with a 'Save Data' button and radio buttons for 'Normal' and 'Compare'. The graph plots two oscillating curves, one blue and one red, over time. The y-axis ranges from 0.2 to 0.6.
- Bottom Panel:** Shows a diagram of the 'student_teacher' capsule. It has two input ports labeled 'teacher rate' and 'student teacher', and one output port labeled 'Mischief Discipline'.

A red circle highlights the 'student_teacher' capsule in the bottom layer, with arrows pointing to the 'student_teacher' text in the middle layer and the 'Mischief Discipline' chip in the bottom layer, illustrating the relationship between the different layers of the model.

What capsules and chips are really useful for is aggregation

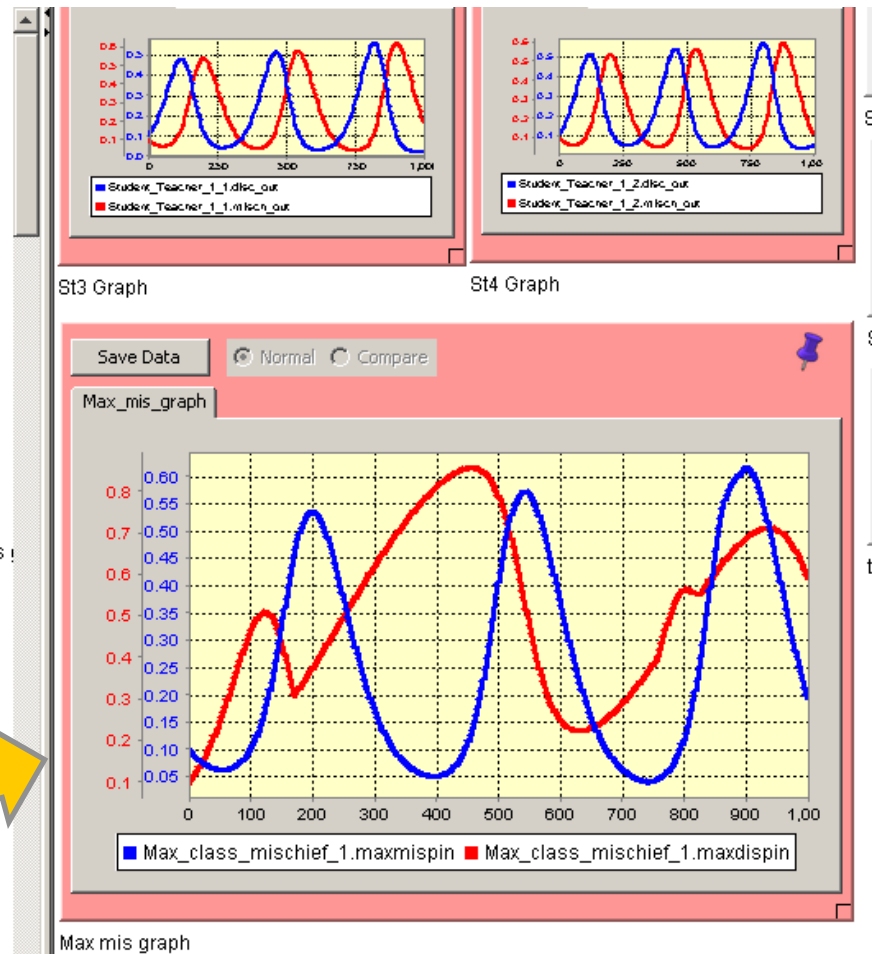
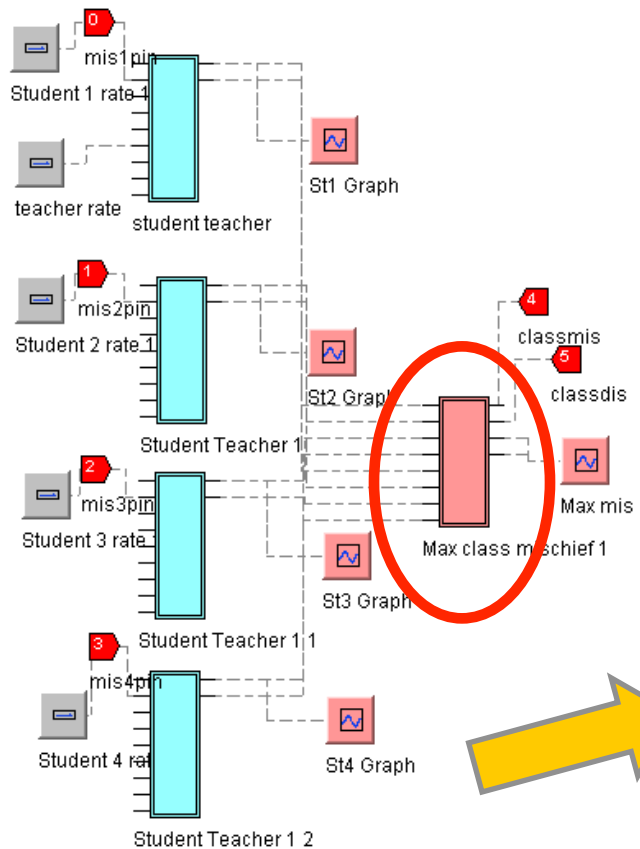
1 mischievous and 3 average students

Four capsules represent 4 students

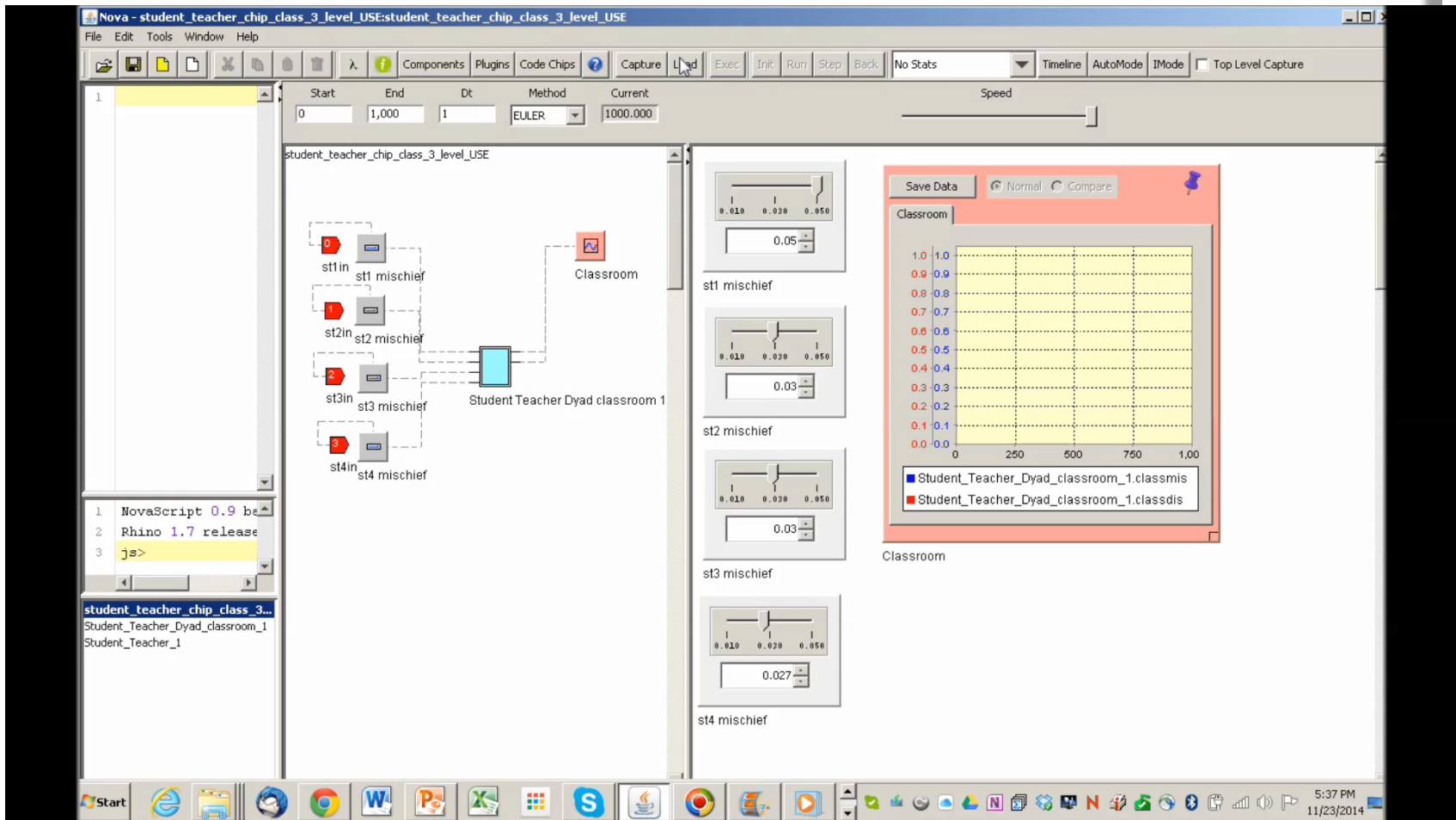
The screenshot displays the Nova modeling environment. The central workspace shows a classroom simulation with four student capsules (labeled 1, 2, 3, 4) and a teacher capsule. Each student capsule is connected to a corresponding graph (St1_Graph to St4_Graph). The graphs show data for 'Student Teacher' and 'Student Teacher' over time. The interface includes a menu bar (File, Edit, Tools, Window, Help), a toolbar, and a status bar at the bottom. The status bar shows the time as 5:01 PM on 11/23/2014. The interface also features several control panels on the right side, including sliders for 'Student 1 rate 1', 'Student 2 rate 1', 'Student 3 rate 1', and 'Student 4 rate', and a 'Max mis' graph.

Another chip takes the output from individual dyads and aggregates them at the classroom level. Now I have a **nested model**.

Student_Teacher_Dyad_classroom_1



And that classroom can be turned into a capsule so it's cleaner



Chips can be configured spatially, so that each student influences the other and the teacher responds to individual and classroom characteristics

The screenshot shows the NOVA modeling environment. At the top, there's a toolbar with icons for file operations and a menu bar with options like 'Components', 'Plugins', 'Code Chips', 'Capture', 'Load', 'Exec', 'Init', 'Run', 'Step', 'Back'. Below the menu bar are parameter controls for 'Start' (0), 'End' (180), 'Dt' (1.0), 'Method' (DISCRETE), and 'Current' (180). A 'Speed' slider is also present.

The main workspace is titled 'Spatial_Max_Instigator'. It contains a diagram with components: 'students' (a red box), 'stat engine 1' (a yellow box), 'teacher owns' (a blue box), 'Mischief STDev' (a grey box), 'Mischief Mean' (a grey box), and 'Mischief' (a purple box). Dashed lines represent connections between these components. A large, multi-pointed starburst graphic is overlaid on the diagram with the text: 'This could also become an agent-based model'.

On the right side of the workspace, there are two data tables. The first table has columns of numerical values, and the second table has columns of numerical values. Below these tables are control panels for 'Dyads' (with 'Normal' and 'Compare' radio buttons), 'Discipline' (with a slider and a value of 50), 'Mischief Mean' (with a slider and a value of 21), and 'Mischief STDev'.

At the bottom left, there is a code editor with the following text:


```

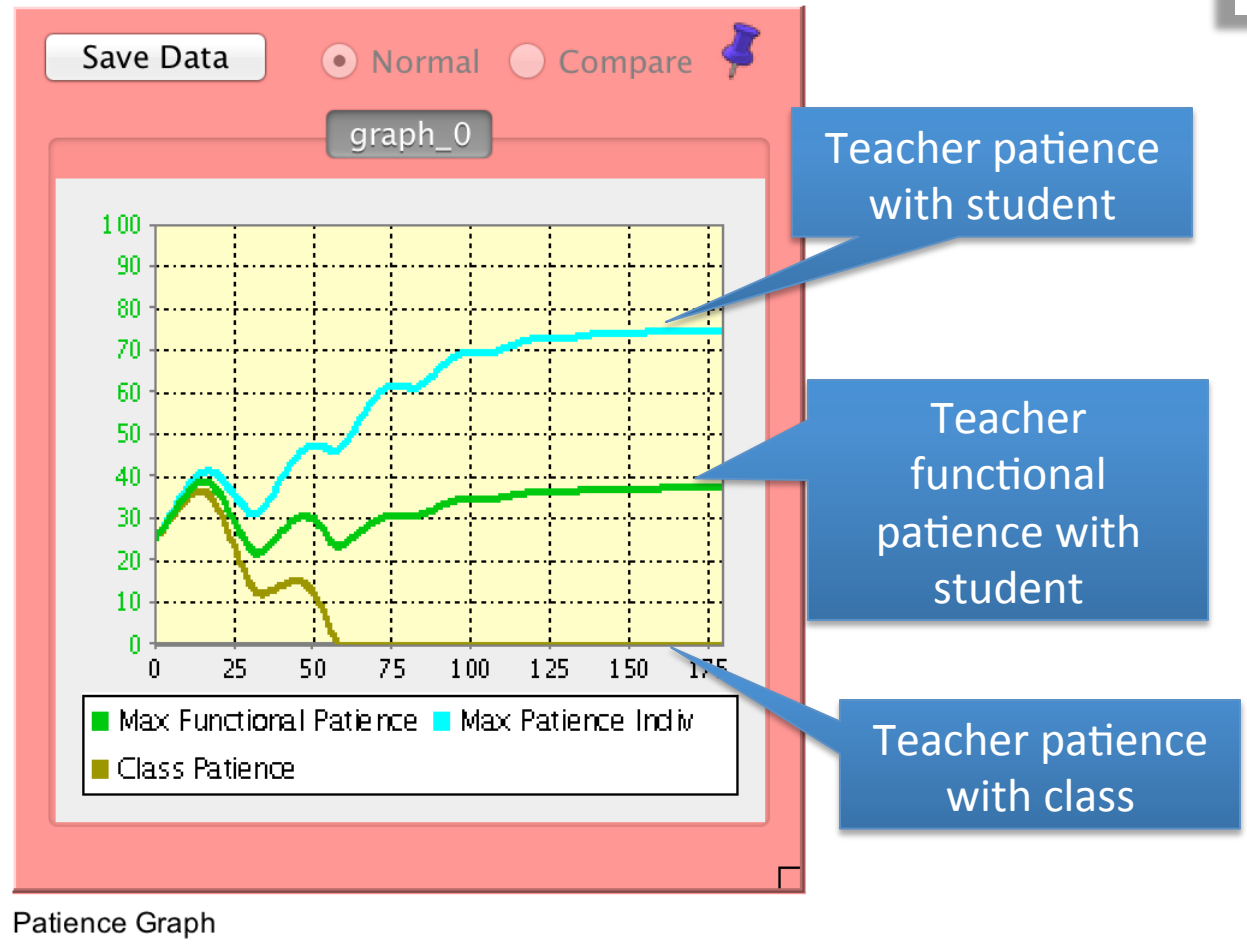
    1 NovaScript 0.9
    2 Rhino 1.7 rele
    3 js>
    
```

 Below the code editor is a list of variables: 'Spatial_Max_Instigator', 'student_owns', 'patience_indiv', 'discipline', 'mischief', 'teacher_owns'.

At the bottom center, there is a text box with the following text:

This is a hypothetical spatial model of a classroom, where a single teacher has a behavioral dyad with every student. In this version the students are affected by their neighbors: their own mischief increases based on their neighbor who is acting out most. The teacher has patience with the class as a whole, as well as a patience with each student reflected

You can model individual dyads within the classroom



Why is that cool? More complex models

- You can take the aggregated classroom mischief and create a stock called 'stress' that decreases a stock called 'patience' that changes the teacher's dyadic reactivity
- You can create contagion effects so each student's behavior changes depending on classroom context
- You could create an agent based model with many more children who find others like themselves and create pockets of mischief through contagion

Running simulations and outputting data

- Nova has the capacity to automatically run through a range of possible values
- Data can be viewed as:
 - Graphs
 - Tables
- Data can be exported as csv or directly to R

Teams Capsules and Modularity: An Example

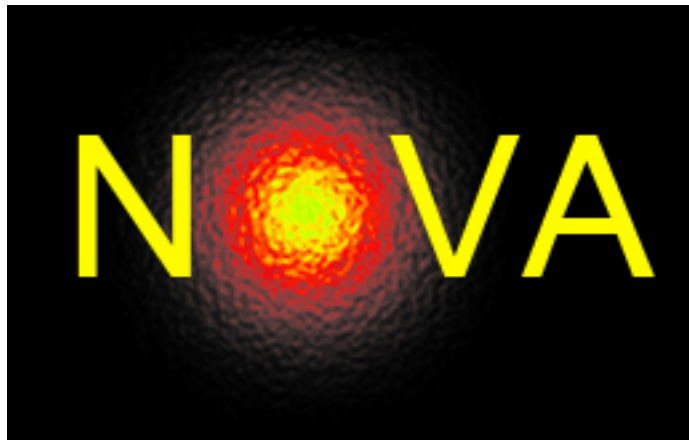
- Nova Online:
 - Multiple stakeholders working to optimize solutions
- Expertise:
 - How do students respond to teachers?
 - How do teachers respond to individual students?
 - How do teachers respond to classroom dynamics?
 - Peer influence on deviant behavior
 - Classroom dynamics

Teams Capsules and Modularity: An Example

- Working with modeling novices: Attachment
- Process:
 - Sketching ideas
 - 3 different model components, 3 different teams
 - Combining and refining
 - Moving from the individual to the couple

Summary

- Nova is a free, flexible program:
 - Stock and flow
 - Spatial
 - Agent based models
- Strengths:
 - One platform for multiple purposes minimizes learning time
 - Nested models
 - Heterogeneous populations with different distributions
 - Good entry-level tutorials
- Weaknesses:
 - Beta
 - Weak documentation of some features

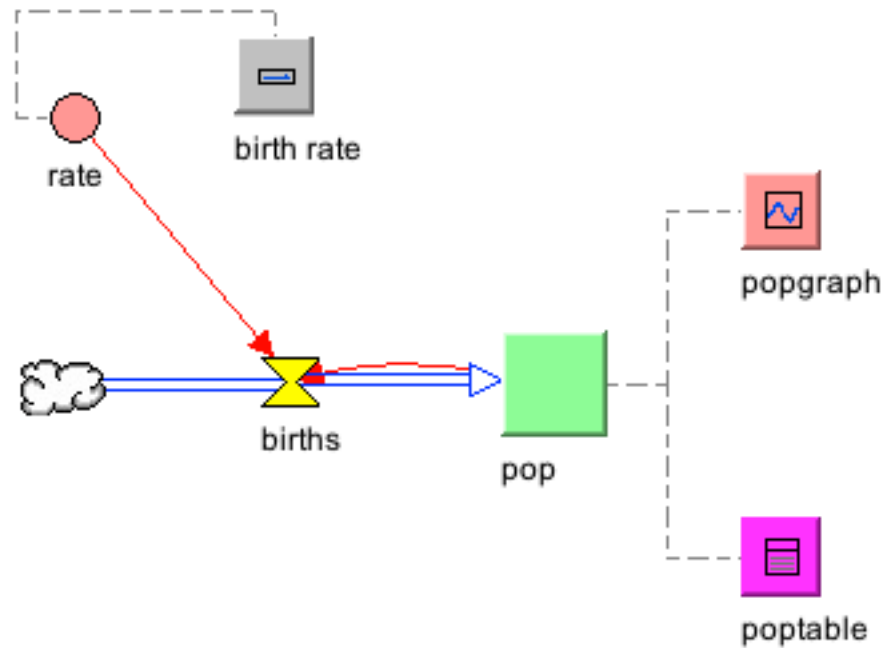


NUTS AND BOLTS

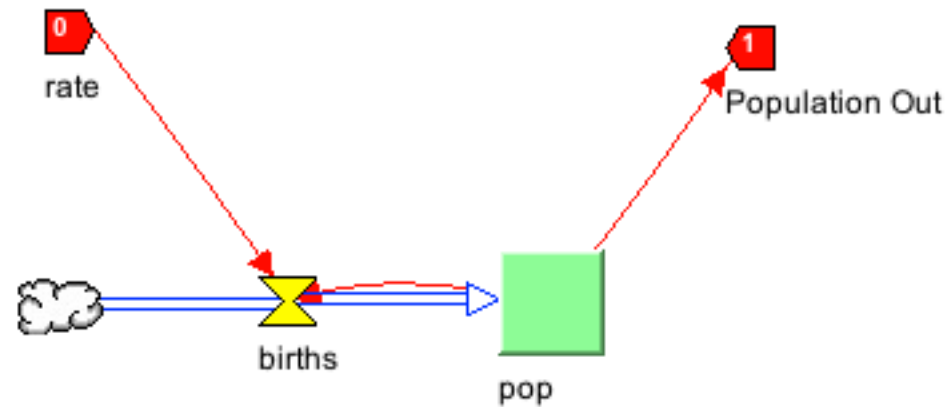
- A single framework for an eclectic set of systems.
- Expressive power derives from
 - modularity
 - abstraction
 - extensibility



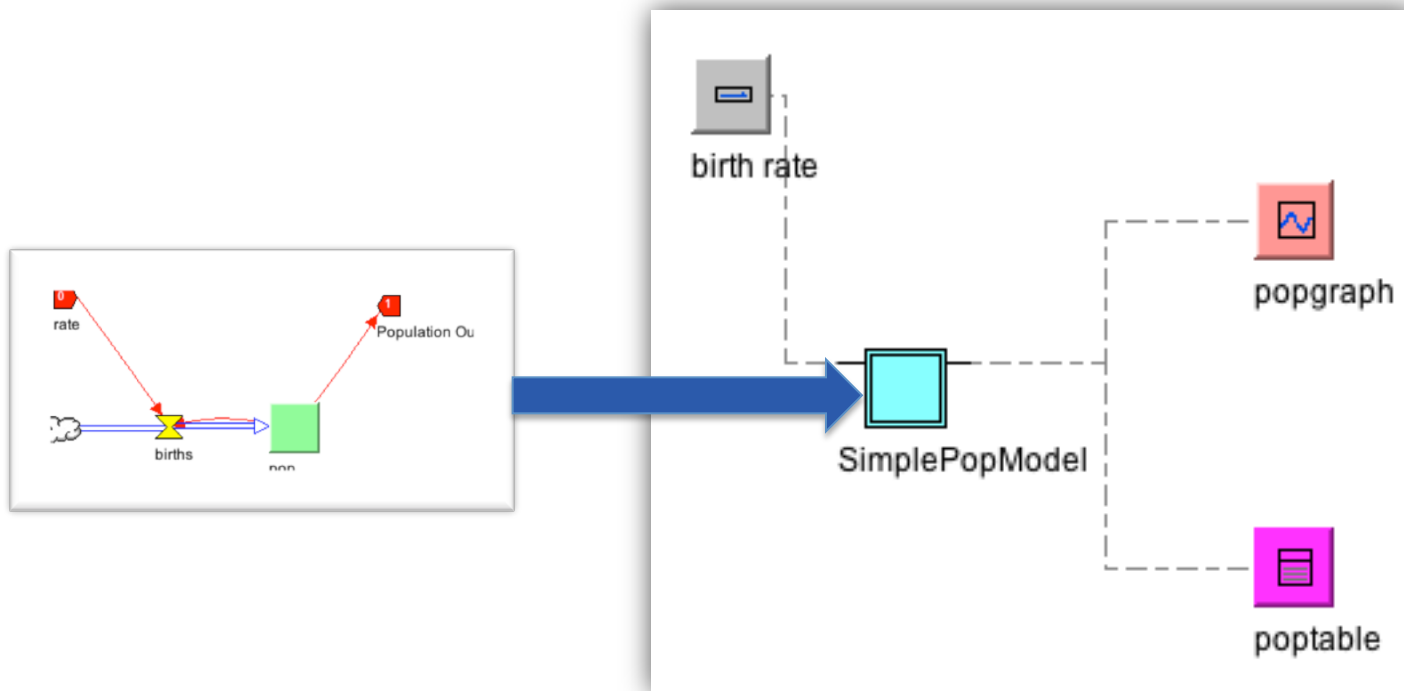
Capsule

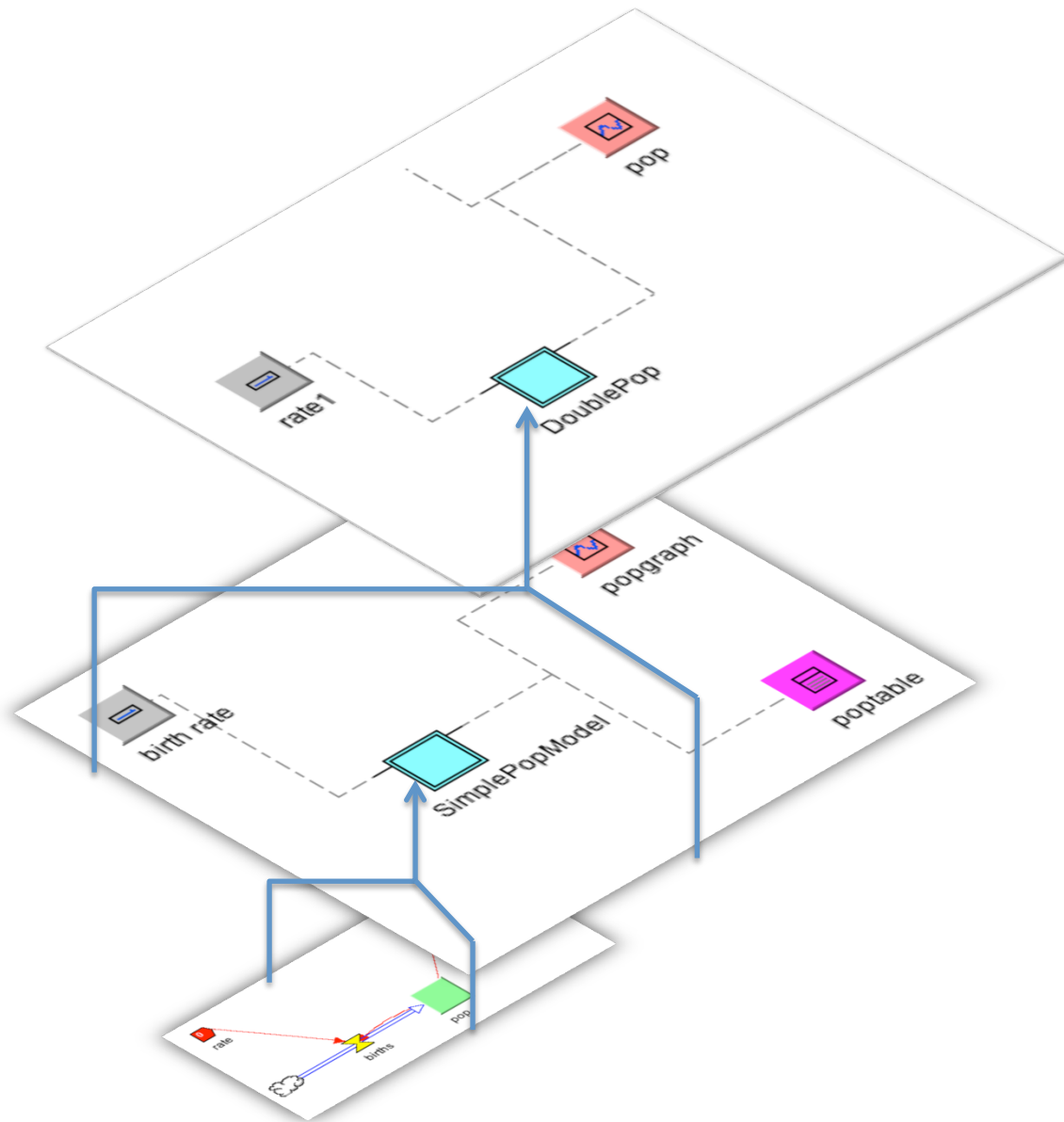


Capsule with Pins



Chip



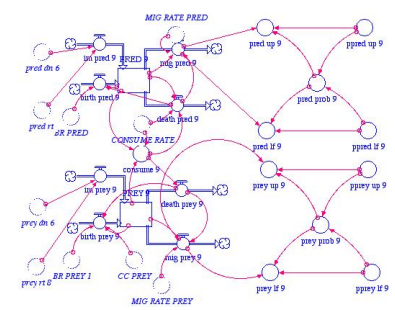
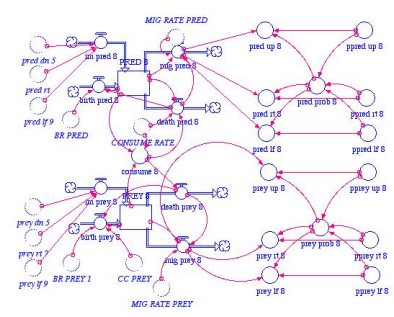
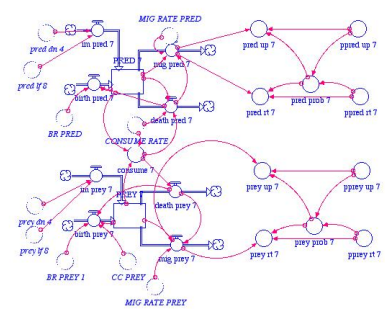
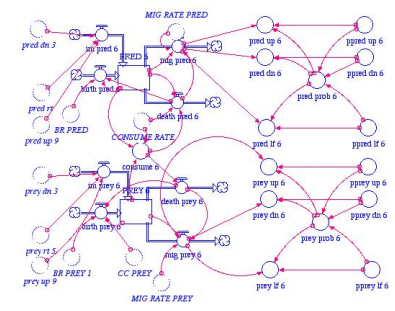
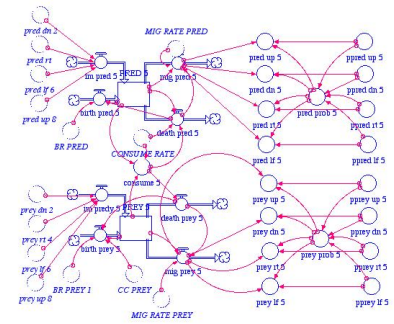
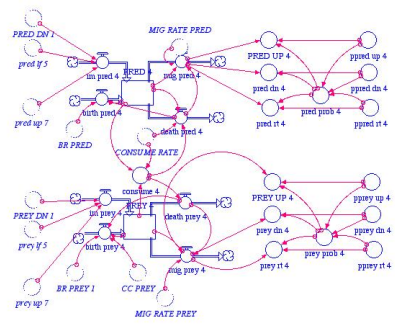
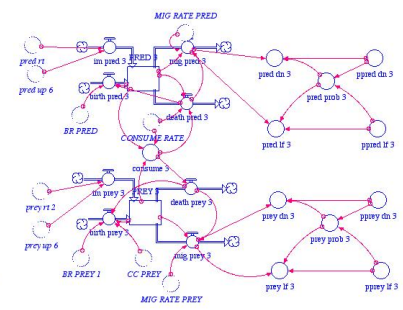
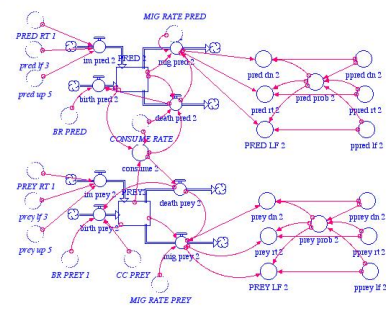
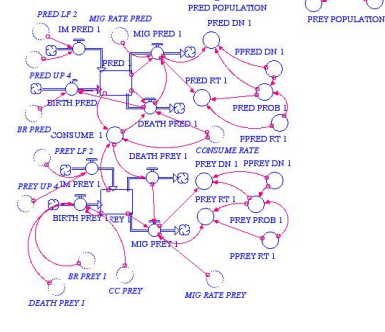
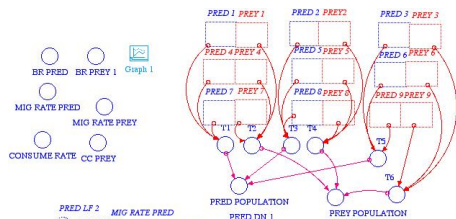


Managing multiple capsules

- Multiple instances of a capsule might be used to represent
 - a spatial grid.
 - a network.
 - a set of agents.

Spatial Modeling in *Stella*

(Hannon, 2001)



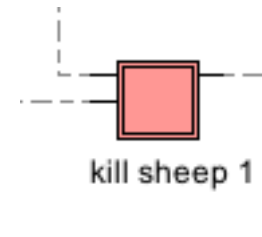
Managing multiple capsules

- Multiple instances of a capsule might be used to represent
 - a spatial grid.
 - a network.
 - a set of agents.
- Nova uses *aggregating components* to manage large sets of individual elements.

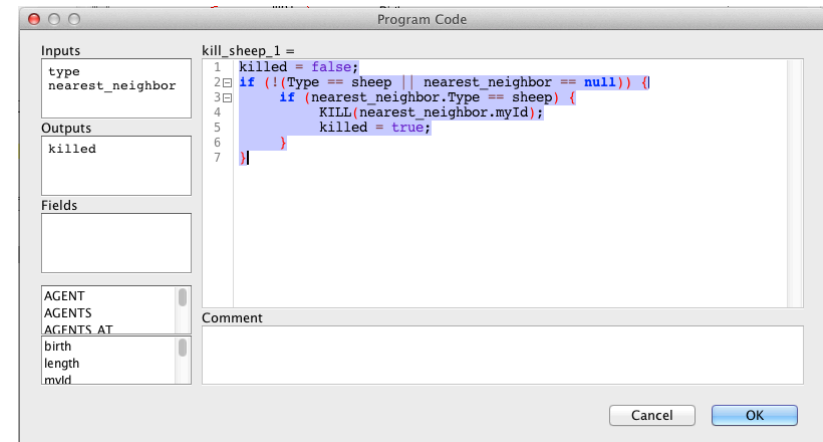
Containers (Aggregators)

- **CellMatrix**
 - 2 dimensional array of capsules
 - facilitates interaction among cells on a Cartesian grid
- **NodeNetwork**
 - An array of capsule *nodes* connected by a set of weighted links (equiv. to a mathematical graph)
 - facilitates transmission of data through the network.
- **AgentVector**
 - Agent = Capsule + location and trajectory parameters
 - AgentVector is 1-dimensional array of agents
 - AgentVector manages a set of agents in a common space
 - spatial position
 - births/deaths
- **SimWorld**
 - CellMatrix + AgentVector
 - Agent space corresponds to Cell topology
 - facilitates interaction between agent and cell environments
- **NetWorld**
 - NodeNetwork + AgentVector
 - Agent space corresponds to Network topology
 - facilitates interaction between agent and node environments

Code Chips



- Contains code implementing a computational method
- Easy to implement multiple instances
- Easy to export/import into new model



A screenshot of a software interface titled "Program Code". The interface has a left sidebar with sections for "Inputs", "Outputs", "Fields", and "AGENT AGENTS AGENTS AT" (with sub-items "birth", "length", "mvid"). The main area contains a code editor with the following code:

```
kill_sheep_1 =
1 killed = false;
2 if (!Type == sheep || nearest_neighbor == null) {}
3   if (nearest_neighbor.Type == sheep) {
4     KILL(nearest_neighbor.myId);
5     killed = true;
6   }
7 }
```

Below the code editor is a "Comment" text area. At the bottom right are "Cancel" and "OK" buttons.

Clocked Chip

- Attach a clock to chip so that each “tick” of the host model corresponds to a complete “run” of the encapsulated model.

Plug-ins

- API for creating new components
- Visualization
- Other useful extensions

Nova Online

- A visual Nova model is “captured” into a script (NovaScript) before it is executed on the Nova runtime engine.
- A Javascript implementation of this runtime has made possible a browser-based runtime using HTML5 graphics:
 - Nova Online
- Currently under construction: automatic creation of Nova Online Website.
- Also under construction: server-side NovaScript runtime for multi-core and high-performance execution.

Collaboration

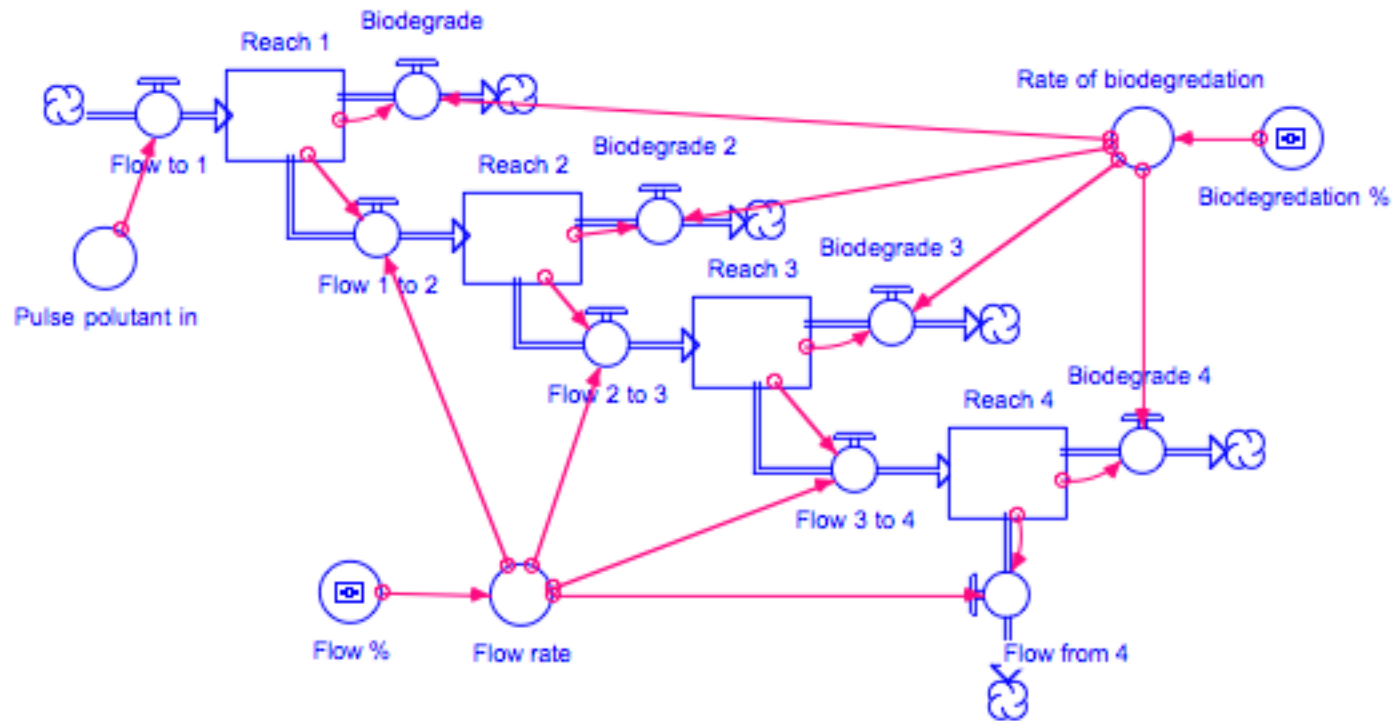
- Sharing of submodels.
- Sharing of codechips.
- Sharing of plug-ins.

- Interaction with R, GIS

- Combining submodels, codechips and plug-ins into a “kit” for a particular application area.

- Nova Website to serve as an archive and marketplace for shared components.

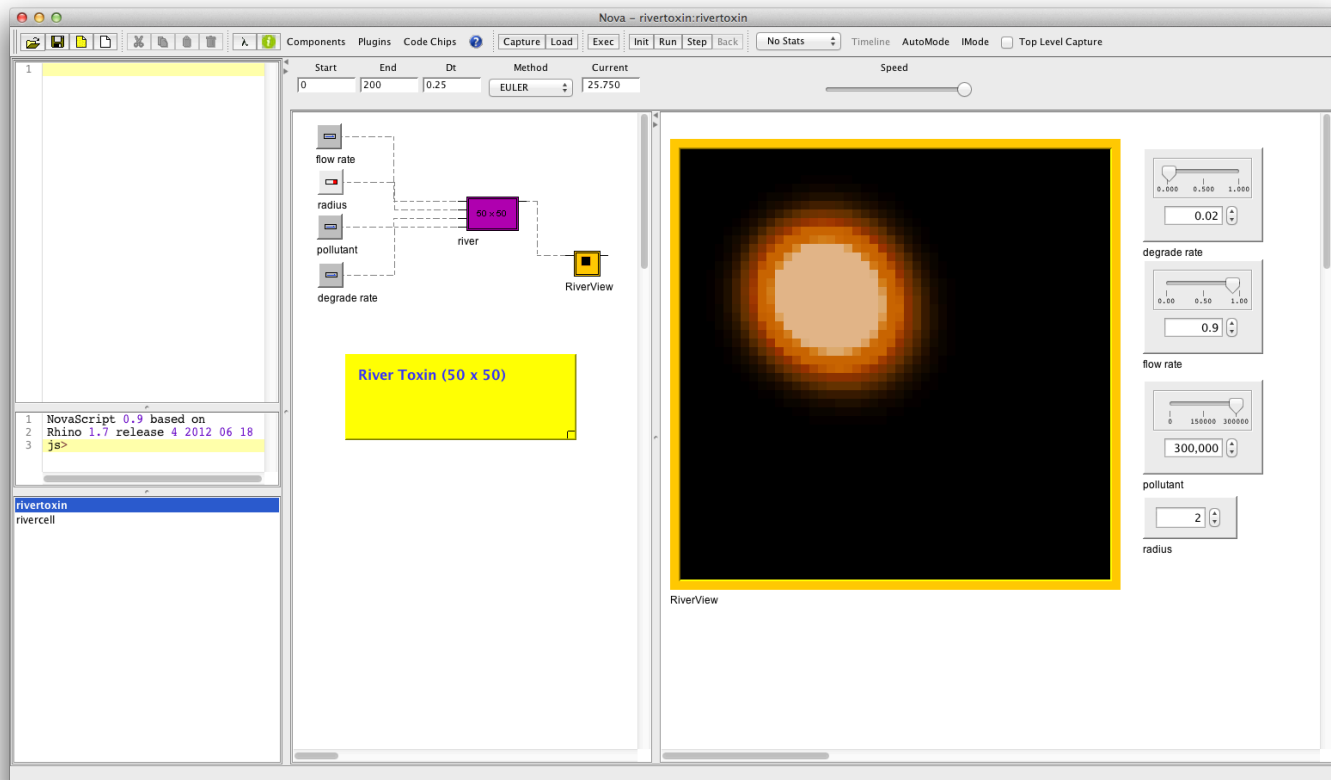
Example 1: River Toxin Advection



STELLA Version

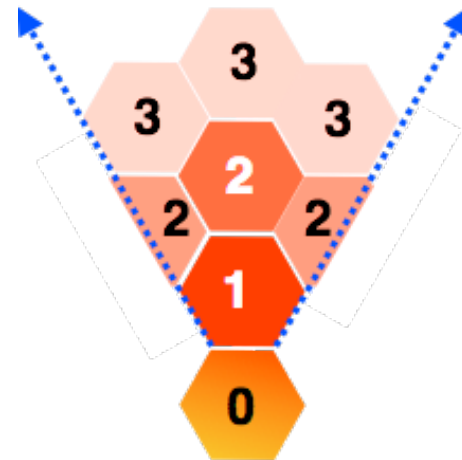
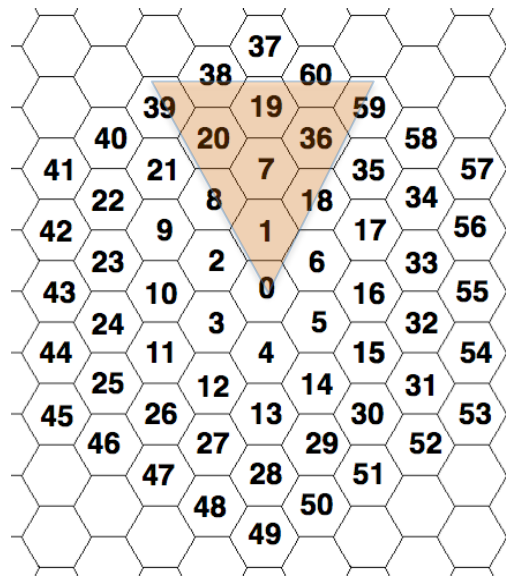
River Toxin: Nova Version

- Model spatially as a grid of cells



Example 2: Hexagonal Grazing (Getz)

- “World” is a hexagonal grid of cells.
- Agents are animals consuming food from cells.
- Cells contain food for consumption.
- At each time step agent must decide to either
 - Eat in current cell
 - Move to an adjacent cell
 - Decision governed by weight parameters: q_1, q_2, q_3, \dots



$$A_1 = q_1 a_1 + q_2 ((a_8 + a_{18})/2 + a_7) + q_3 (a_{19} + a_{20} + a_{36})$$

Example 3:

Florida invasive snail – *Pomacea maculata*

- Model depicts a 25 square meter area with patches of size 10^{-2} sq m. Four snail "types" shown:
 - Males (blue)
 - Unfertilized Females (pink)
 - Fertilized Females (red)
 - Juveniles (yellow)
- Once fertilized, female lays an eggcase with up to 1000 eggs every 14 days (laying action depicted as enlarged purple agent token). Eggs hatch in 14 days with a 10% survival rate.
- Juveniles mature to adult status in 120 days (size of juvenile agent token grows with age).
- Separate juvenile/adult movement and consumption rates used.
- Attraction of males to unfertilized females is modeled.
- Carrying capacity is proportional to current biomass.
- Five year timespan modeled with seasonal variation of biomass growth.
- Snail aestivation occurs in December and January.
- Actual GIS-derived terrain is depicted.

www.novamodeler.com

Nova

