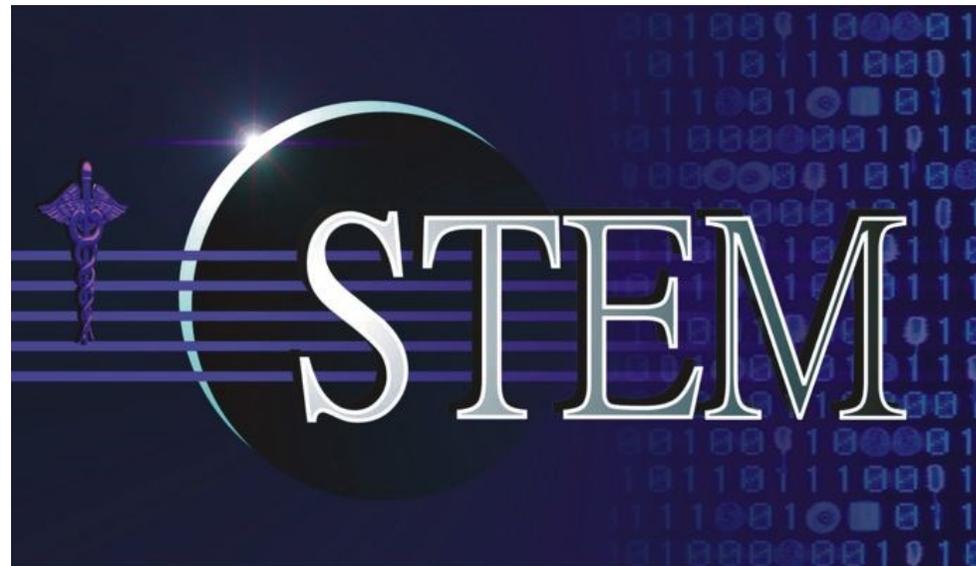
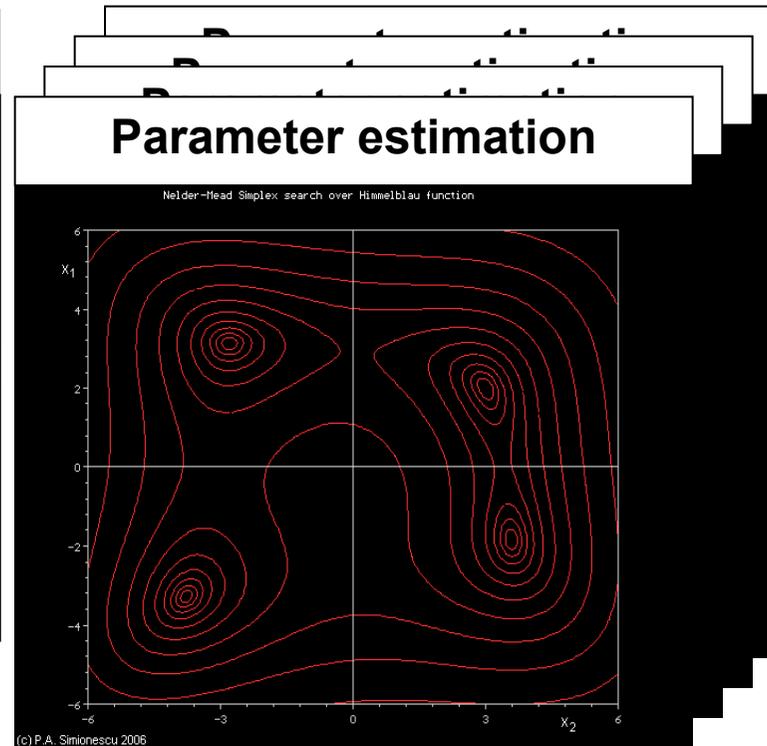
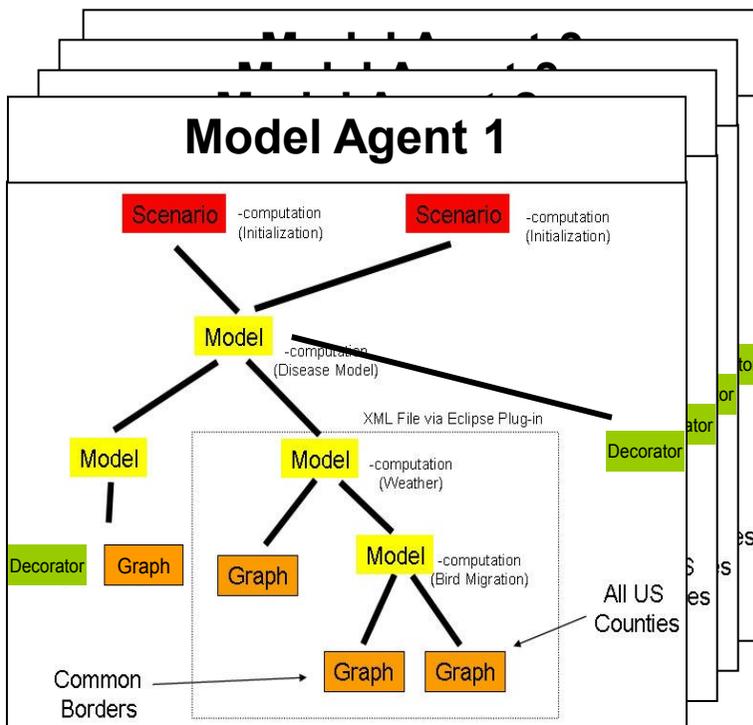


# STEM - Spatiotemporal Epidemiological Modeler: Overview on Advanced Features

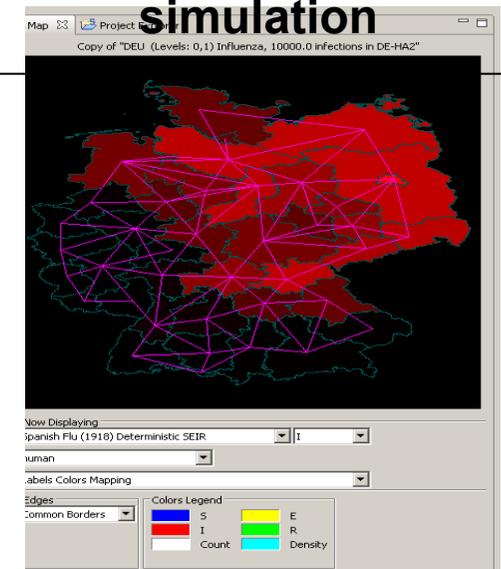


Matthias Filter

## Knowledge Engineer / Statisticians

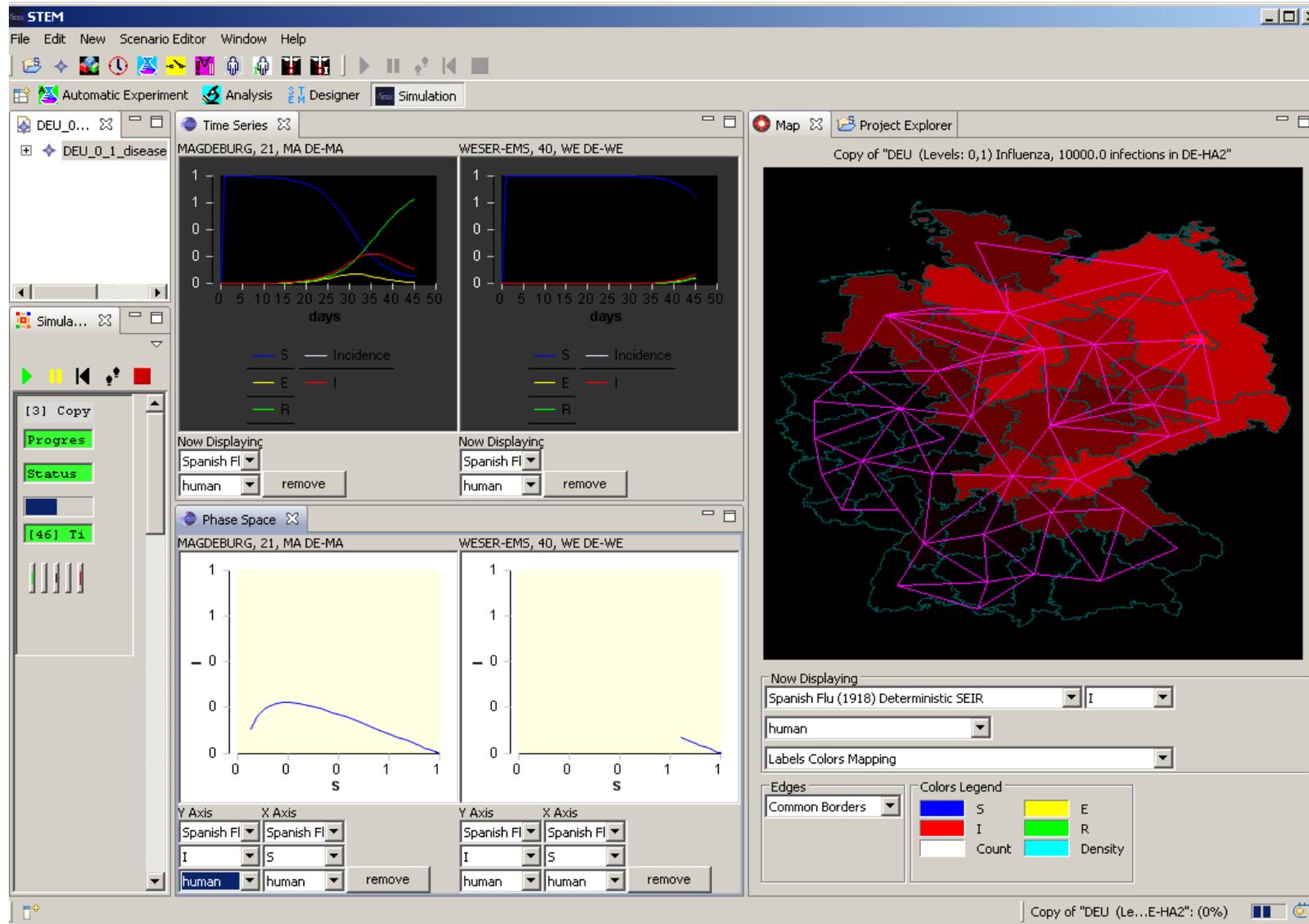


**Library of models for visualization and simulation**



**Public Health Risk Assessor / Manager**

# GUI for simulation, visualization and analysis



# Integrated Data supporting Global Simulations



Global population data by admin region validated against *LandScan 2007*<sup>TM</sup>

*High Resolution Global Population Data Set*

© UT-Battelle, LLC, operator of Oak Ridge National Laboratory

# GUI for scenario and model generation



The screenshot displays the STEM software interface. On the left is the 'Project Explorer' showing a hierarchical view of various models and scenarios. The main workspace shows a tree view for the 'Contam.scenario Scenario', including sub-models like 'dis.model Model', 'pop.model Model', and 'geo.model Model'. At the bottom, a 'Properties' table is visible for the selected object.

Property	Value
Add Stochastic Noise	<input type="checkbox"/> False
Characteristic Mixing Distance	<input type="text" value="2.25"/>
Disease Name	<input type="text" value="UnknownContamination2"/>
Frequency Dependent	<input checked="" type="checkbox"/> true
Immunity Loss Rate	<input type="text" value="1.0"/>
Infectious Mortality Rate	<input type="text" value="0.0"/>
Non Linearity Coefficient	<input type="text" value="1.0"/>
Population Identifier	<input type="text" value="human"/>
Random Seed	<input type="text" value="1"/>
Recovery Rate	<input type="text" value="0.4"/>
Reference Population Density	<input type="text" value="100.0"/>
Road Network Adjacent Infectious Proportion	<input type="text" value="0.01"/>
Time Period	<input type="text" value="86400000"/>

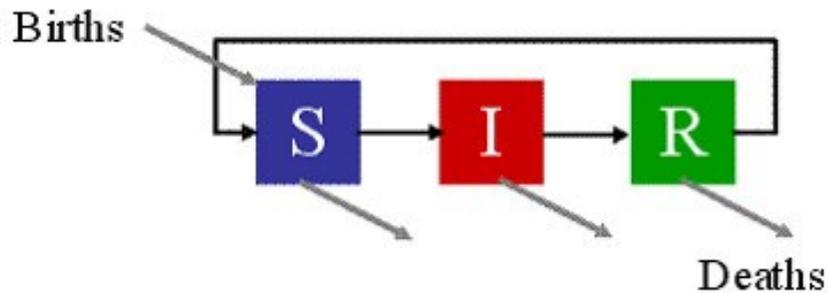
Selected Object: UnknownContamination2.standard SIR Compartment Model

# Basis I: Epidemiological Models



- Deterministic and stochastic simulations including pre-built compartment models (e.g. SIR, SEIR etc.)
- Code generation tool for new models

## Standard SIR model



[Int J Health Geogr.](#) 2006 Jan 17;5:4.  
An extensible spatial and temporal epidemiological modelling system.

## Model Equations

$$\Delta S = -\beta (S/P) I + \alpha R + \mu (P-S)$$

$$\Delta I = \beta (S/P) I - \gamma I - \mu I$$

$$\Delta R = \gamma I - \alpha R - \mu R$$

$$\text{Total Population } P = (S + I + R)$$

## Model compartments:

S: Susceptible

I: Infectious

R: Recovered

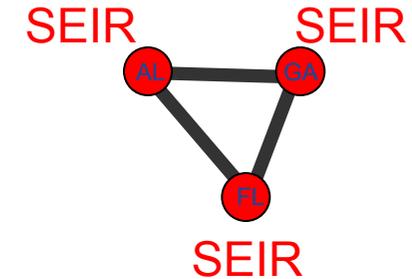
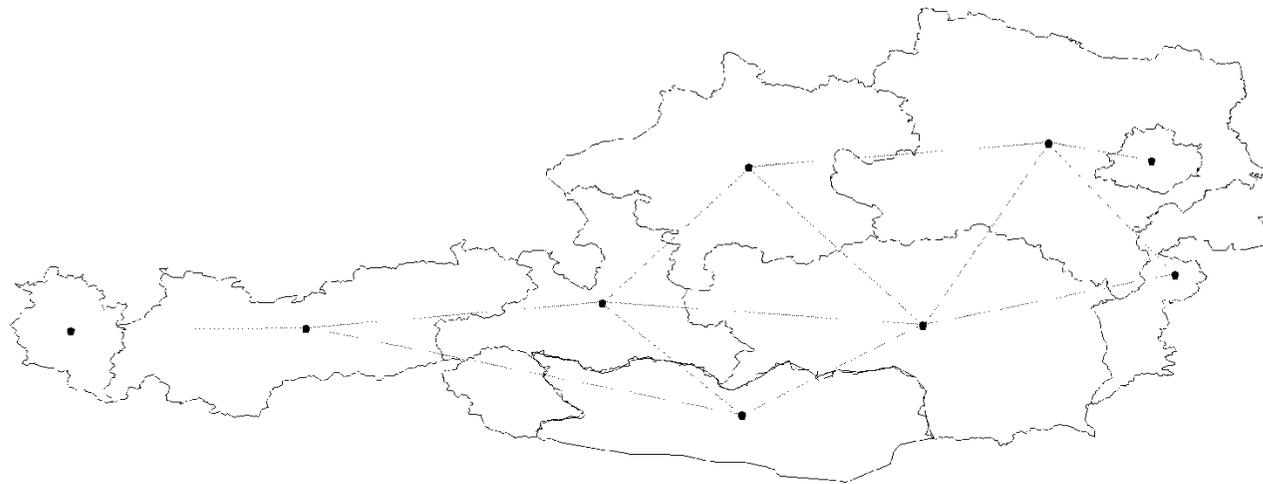
## Model parameters:

- Transmission rate  $\beta$
- Recovery rate  $\gamma$
- Immunity loss rate  $\alpha$
- Birth/death rate  $\mu$

# Basis II: Spatial distribution of diseases



## STEM treats the World as a “Graph”



- Graph = nodes, edges; with labels & decorators
- Graph Framework makes it easy to *build one model on top of another*
- Existing plug-ins define
  - Geography
  - Populations
  - Transportation systems
  - Land area

# Basis III: Canonical Graphs (Integration of all components)



**Scenario:** Model + Infector + Sequencer

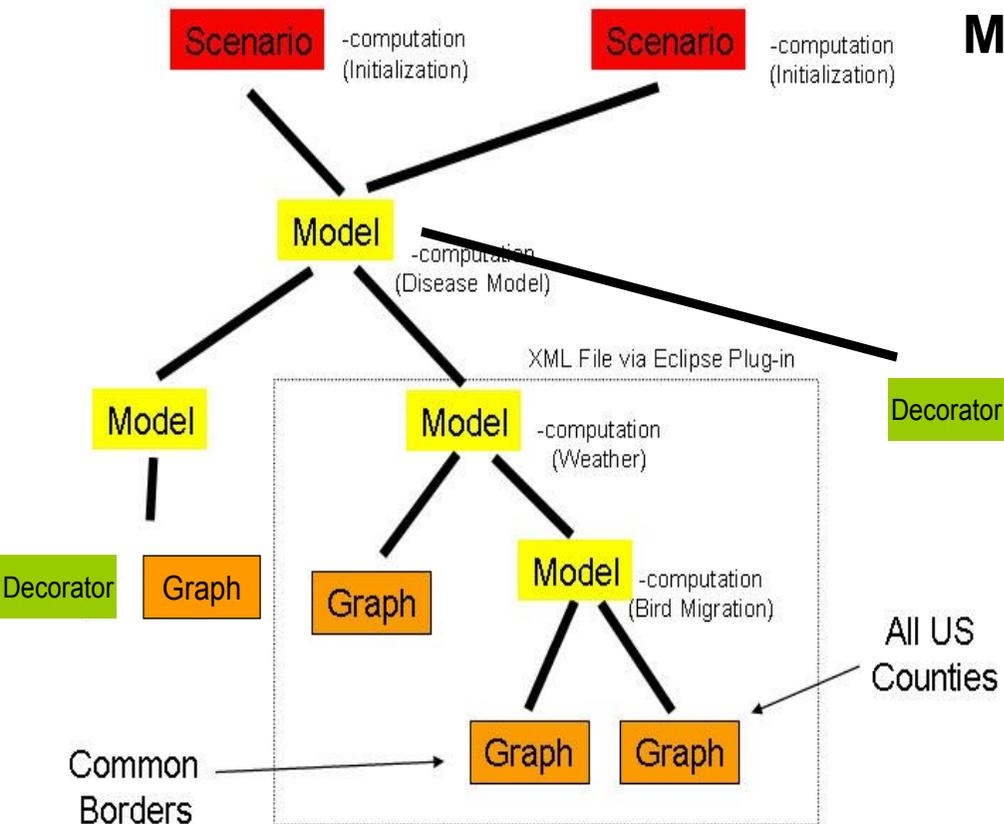
**Model:** Graphs + Decorators

**Graphs:** Nodes (Labels) + Edges (Labels)  
 may comprise several subgraphs  
 i.e. places, regions  
 i.e. relationships, connections  
 between nodes, vectors

**Decorators:** denominator data and model  
 parameters  
**Disease:** Disease model parameter  
**Modifier:** Parameter tuning components

**Infector:** Disease initiation parameters

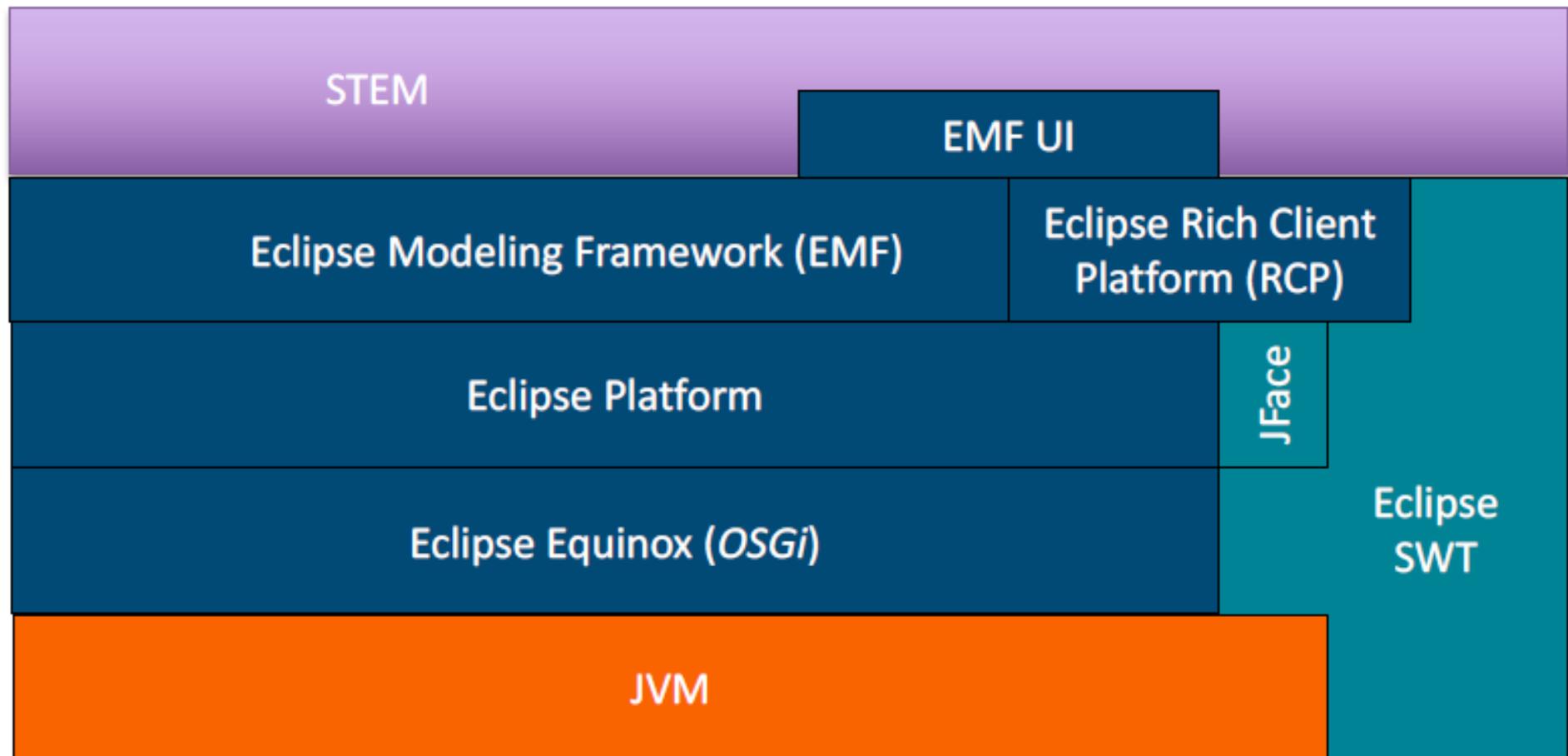
**Sequencer:** Simulation time parameter



# Basis IV: STEM Software Architecture (and language)



- Built on Eclipse Platform and Modeling Framework, SWT/JFace/RCP UI



Courtesy by Kun Hu, Stephan Edlund, James Kaufman et al., IBM Research,

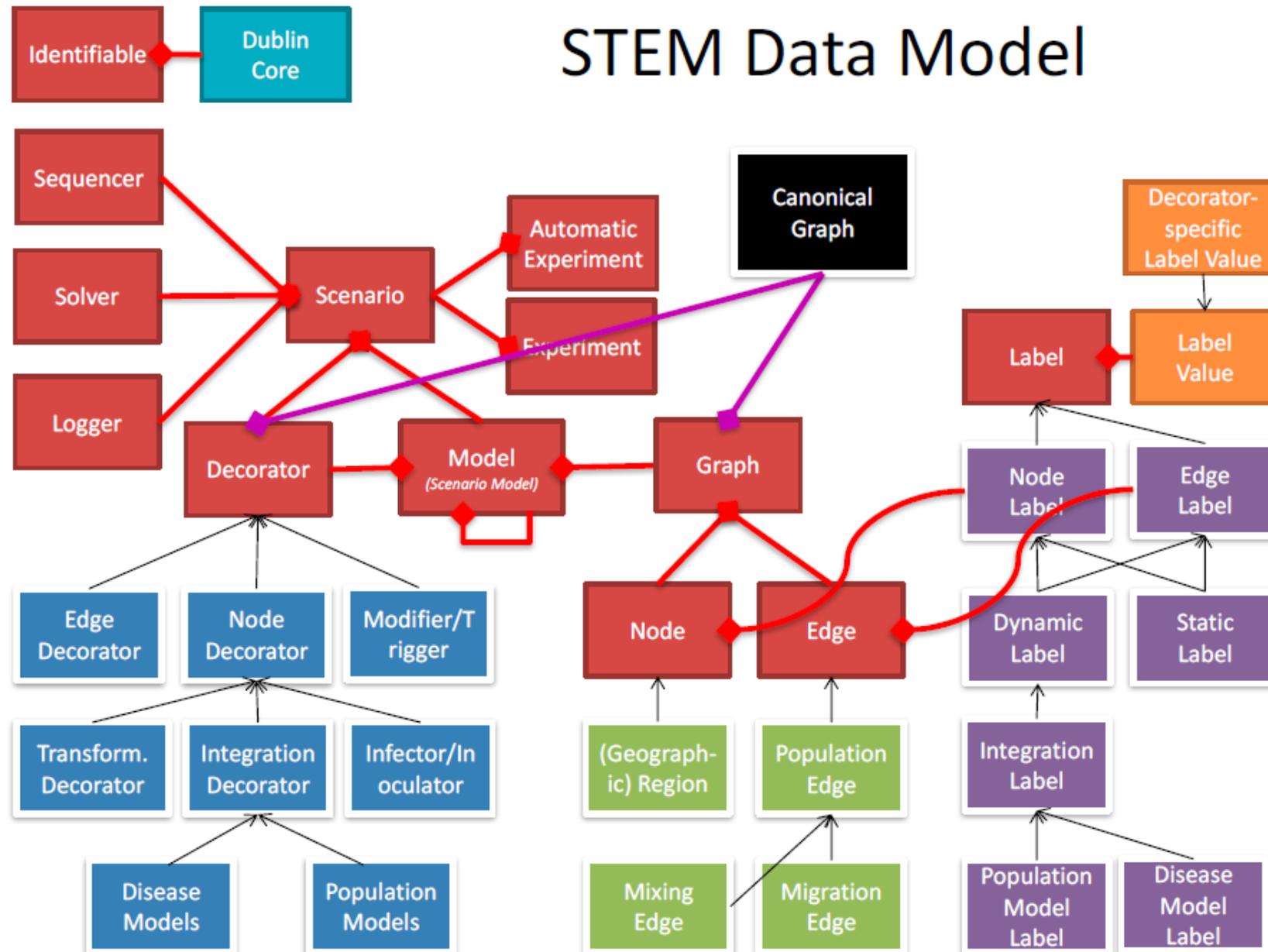
# STEM Terminology (“What is meant by “Model””)



- We use the word “Model” interchangeably
- Depending upon context, it’ll be related to one of the following:
  - Data modeled, code-generated data structures (**STEM Data Model or EMF Models**)
  - Container in STEM UI for composing scenarios (**Scenario Model or Geographic model**)
  - Dynamic, integrated, programmatic models (**Disease or population model**, like Influenza)
  - **Model builder** – tool for authoring disease models

Courtesy by Kun Hu, Stephan Edlund, James Kaufman et al., IBM Research,

# Basis V: STEM Data Model



Courtesy by Kun Hu, Stephan Edlund, James Kaufman et al., IBM Research,

# STEM Components



- Core
  - Core data (EMF) models
    - Identifiable, Scenario, Model, Graph (Node/Edge), Decorators, Labels, ...
  - Simulation Engine ([org.eclipse.stem.jobs](http://org.eclipse.stem.jobs))
  - Solver, sequencer, logger, graph generator implementations
- Data
  - Geography, Population, Transportation, Earth Science
  - Tools for generating fixed plug-ins of data components (“built-in data”)
- Epidemiological models
  - Disease Model implementations (SI, SIR, SEIR, Multi-population, Malaria, Polio, etc)
  - Sample diseases
- Population models
  - Dynamic population models (e.g. mosquito population)
- Food Production models
- Model Generator
- User Interface for all of the above

Courtesy by Kun Hu, Stephan Edlund, James Kaufman et al., IBM Research,

# STEM Canonical Graph



- The runtime representation of a simulation's state
  - *simulation.getScenario().getCanonicalGraph()*
- Generated from the Scenario when a simulation is started
  - Flattened and resolved version of the combined Graphs supplied to the Scenario
- Contains
  - Decorators
  - Nodes
  - Edges
  - Time

Courtesy by Kun Hu, Stephan Edlund, James Kaufman et al., IBM Research,



- Objects that act on the canonical graph by **creating** and **updating** Labels
  - Decorators “decorate” the graph
- Can act on the graph, nodes, or edges
  - Applied to the (geographic) model or scenario
- Decorator examples
  - Disease models (Integration Decorator)
  - Dynamic population models
  - Infectors, inoculators
  - Triggers, modifiers

Courtesy by Kun Hu, Stephan Edlund, James Kaufman et al., IBM Research,



- Objects containing simulation state data attached to nodes and edges
- Both static (fixed values) and dynamic (variable values)
  - Integration labels provide for numerical integration of current and next values
- Labels contain one or more **LabelValue** objects with the label's property values

Courtesy by Kun Hu, Stephan Edlund, James Kaufman et al., IBM Research,



## Static Labels

- Node Labels
  - Census human population
  - Area
  - Elevation
  - Earth Science
    - Temperature, nighttemp, relative humidity
- Edge labels
  - Physical Relationships
    - Common Border
    - Physical Containment
  - Migration rate between nodes
  - Transportation rate

## Dynamic Labels

- Dynamic Node Labels
  - Disease model compartment states
  - Dynamic population models
    - E.g. vector-borne

Courtesy by Kun Hu, Stephan Edlund, James Kaufman et al., IBM Research,

# STEM Labels – How they work



- A disease model provides a **Label** and **LabelValue**
- A STEM simulation contains
  - A single instance of the disease model (Decorator)
  - A label for each region in geographic model
- A disease model's label contains *current* and *next LabelValue* objects that store the compartment states for the node
  - Example: A SIR disease model's LabelValues contains s, i, and r properties
- The Solver calls *calculateDeltas(...)* on the Decorator, which is responsible for calculating the Label's *nextValue*
- The solver integrates between the values

Courtesy by Kun Hu, Stephan Edlund, James Kaufman et al., IBM Research,



- Advanced Population Initialization (Structured Populations, Aging Populations)
- Advanced Disease Initialization (Infectors, Inoculators, and Initializers)
- Working with Graphs
  - Composing a Graph
  - Creating a Custom Graph
  - Visualizing and Editing Graphs with the STEM Graph Editor
  - Importing a Graph from a Pajek File
  - Importing a Graph from an Esri Shapefile
  - Import Discrete Transportation Events
- Triggering Interventions
- Running Experiments (in batch mode)
- Invoking Modifiers Using Triggers and Predicates.



- Logging Data to Files / Importing Data from Files
- STEM Model Creator
- Running an Automated Experiment / STEM Analysis Perspective
  - Estimating Model Parameters from External Data
  - Epidemic Analysis
  - RMS Comparison Between Data Sets
  - Lyapunov Analysis
- Running STEM Headless
- Etc.

# Working with Graphs

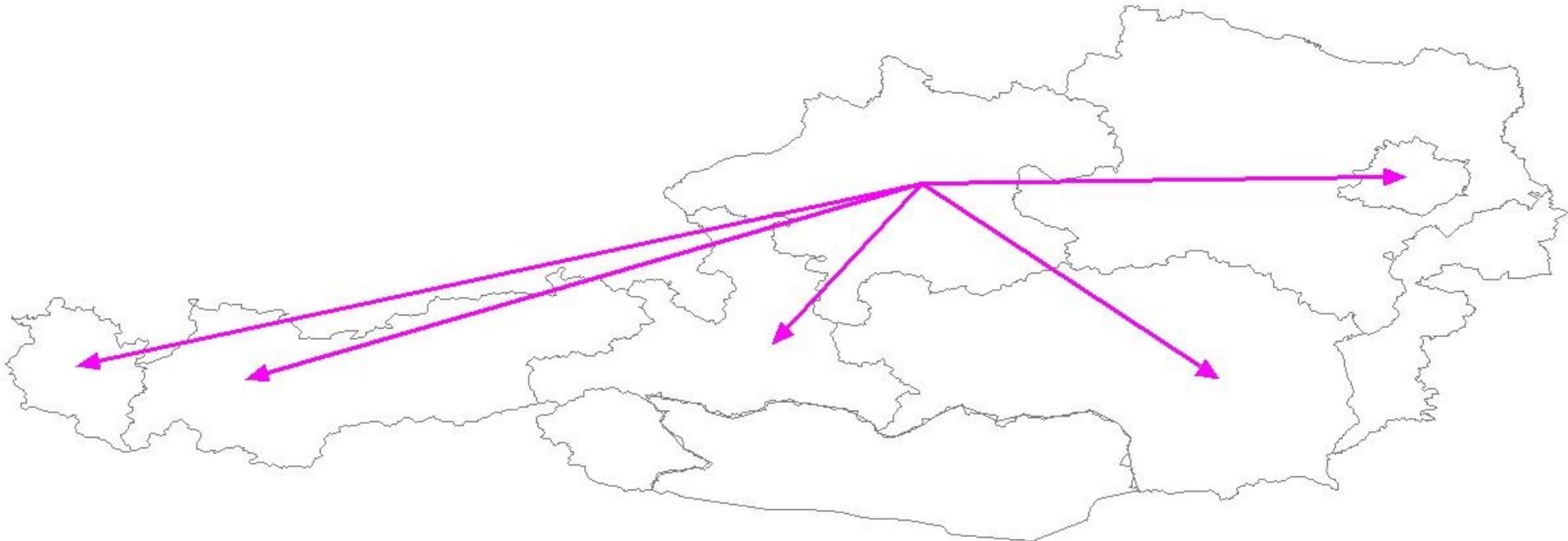
## Pajek Net Graph Generator



Pajek-like format

Define movement (disease vectors,  
travel, transportation events,)

```
*Vertices 9
1 AT-10
2 AT-2
3 AT-3
4 AT-4
5 AT-5
6 AT-6
7 AT-7
8 AT-8
9 AT-9
*Edges
1 2 popID beef rate 0 date 01.01.12 rate 10
1 3 popID beef rate 0 date 01.01.12 rate 10
1 4 popID beef rate 0 date 01.01.12 rate 10
1 5 popID beef rate 0 date 01.01.12 rate 10
1 6 popID beef rate 0 date 01.01.12 rate 10
```



# Pajek Net Graph Generator



## Open Office Macro

pajek\_beef.ods - OpenOffice.org Calc

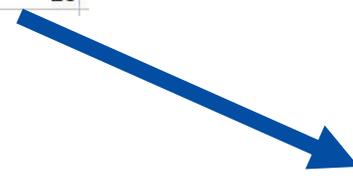
File Edit View Insert Format Tools Data Window Help

B21

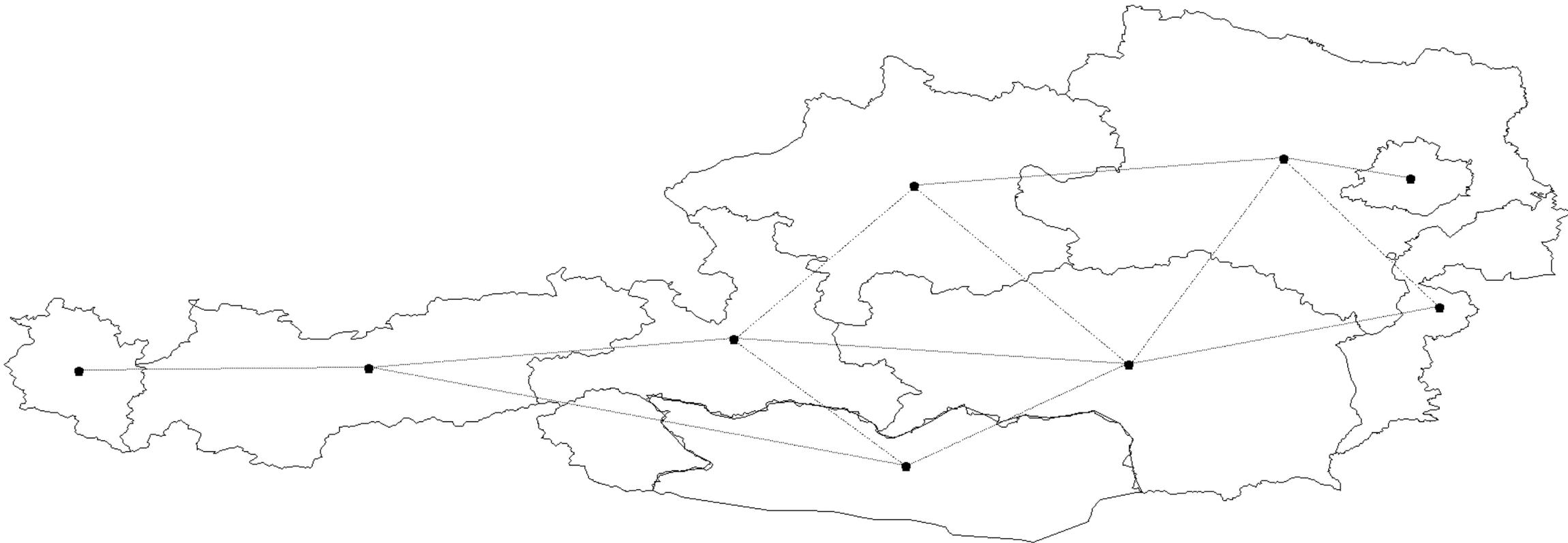
	A	B	C	D	E	F
1	NodeA	NodeB	PopulationID	Rate	Date	Rate
2	1	2	beef	0	01.01.12	10
3	1	3	beef	0	01.01.12	10
4	1	4	beef	0	01.01.12	10
5	1	5	beef	0	01.01.12	10
6	1	6	beef	0	01.01.12	10
7	1	7	beef	0	01.01.12	10
8	1	8	beef	0	01.01.12	10
9	1	9	beef	0	01.01.12	10
10	2	3	beef	0	02.01.12	20
11	2	4	beef	0	02.01.12	20
12	2	5	beef	0	02.01.12	20
13	2	6	beef	0	02.01.12	20
14	2	7	beef	0	02.01.12	20
15	2	8	beef	0	02.01.12	20
16	2	9	beef	0	02.01.12	20

```

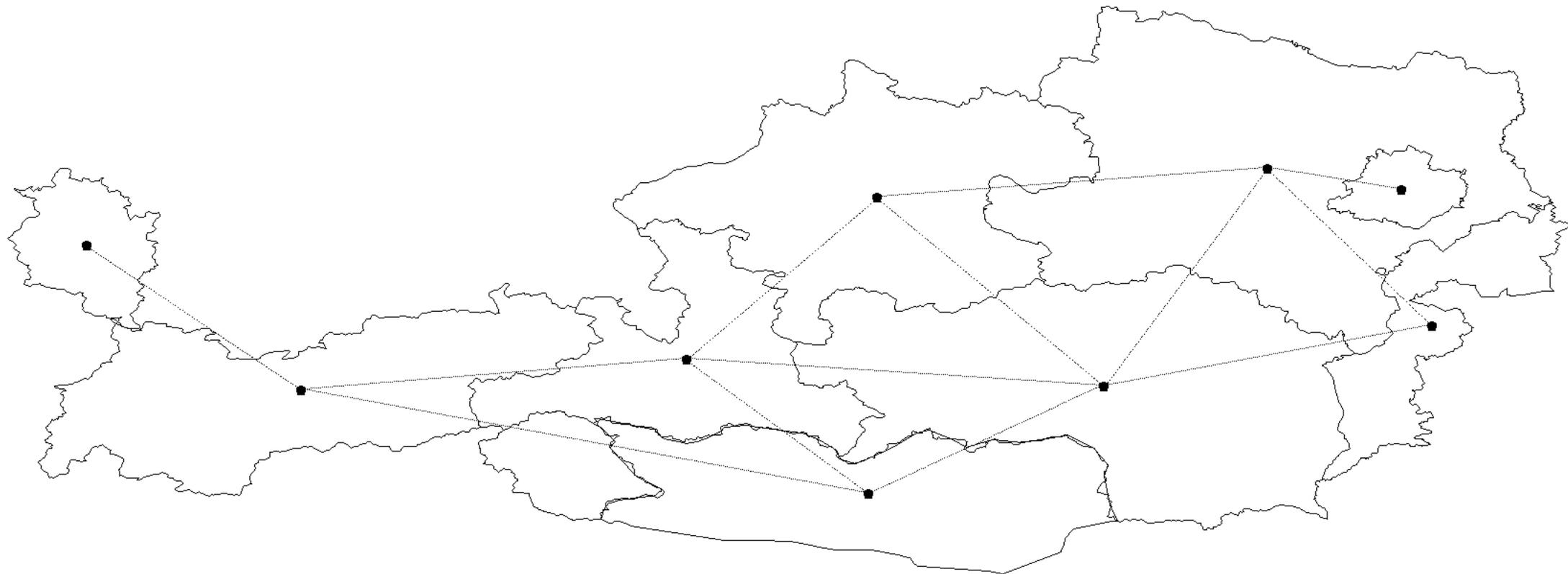
*Vertices 9
1 AT-10
2 AT-2
3 AT-3
4 AT-4
5 AT-5
6 AT-6
7 AT-7
8 AT-8
9 AT-9
*Edges
1 2 popID beef rate 0 date 01.01.12 rate 10
1 3 popID beef rate 0 date 01.01.12 rate 10
1 4 popID beef rate 0 date 01.01.12 rate 10
1 5 popID beef rate 0 date 01.01.12 rate 10
1 6 popID beef rate 0 date 01.01.12 rate 10
1 7 popID beef rate 0 date 01.01.12 rate 10
1 8 popID beef rate 0 date 01.01.12 rate 10
1 9 popID beef rate 0 date 01.01.12 rate 10
2 3 popID beef rate 0 date 02.01.12 rate 20
2 4 popID beef rate 0 date 02.01.12 rate 20
2 5 popID beef rate 0 date 02.01.12 rate 20
2 6 popID beef rate 0 date 02.01.12 rate 20
2 7 popID beef rate 0 date 02.01.12 rate 20
2 8 popID beef rate 0 date 02.01.12 rate 20
2 9 popID beef rate 0 date 02.01.12 rate 20
    
```



# Graph Editor



# Graph Editor



# Graph Editor

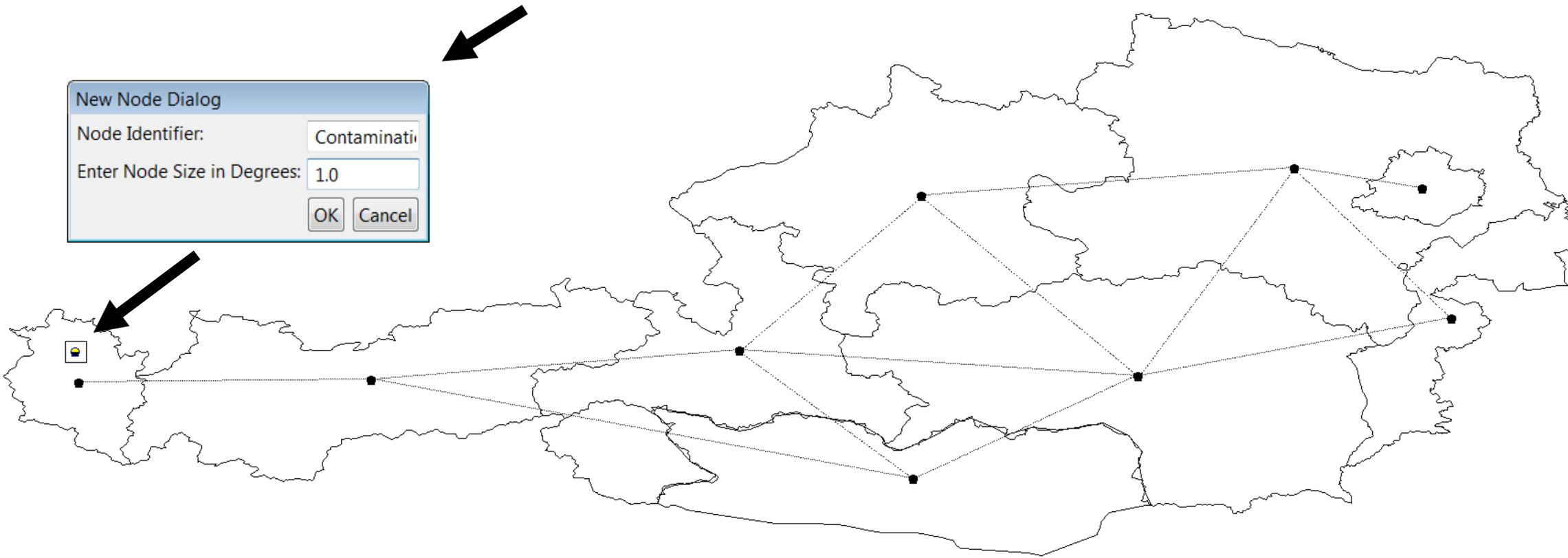


Add Node

New Node Dialog

Node Identifier:

Enter Node Size in Degrees:



# STEM Model Creator



The screenshot displays the STEM Model Creator software interface. On the left, a sidebar contains a tree view with nodes for "Model Properties", "Model Compartments", and "Model Parameters Page". The "Model Parameters Page" is currently selected and expanded, showing a list of parameters with columns for "Name" and "Value". The parameters listed include: transmissionRate2, populationIdentifier, timePeriod, diseaseName, finiteDifference, frequencyDependent, referencePopulationDensity, roadNetworkInfectiousProportion, characteristicMixingDistance, transmissionRate, nonLinearityCoefficient, recoveryRate, infectiousMortalityRate, and immunityLossRate. Below the list are buttons for "Add Parameter" and "Edit Parameter".

In the foreground, a code editor window titled "CattleSalmonellaImpl.java" is open, displaying the following Java code:

```
/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
@Override
public void calculateDeltas(STEMTime time, double t, long timeDelta, EList<DynamicLabel> labels) {
    // TODO Auto-generated stub. Implement calculateDeltas(...) for CattleSalmonellaImpl.

    // Add common calculations here that needs to be done before we begin.
    // For instance, adjust rate parameters here to take into account the time
    // step of the sequencer and the time period of the disease model.
    // Here is an example:
    // final double adjustedTransmissionRate = getTransmissionRate() * ((double) timeDelta / (double)

    for(int _i=0;_i<labels.size();++_i) {
        CattleSalmonellaLabel diseaseLabel = (CattleSalmonellaLabel)labels.get(_i);
        CattleSalmonellaLabelValue currentDiseaseState = (CattleSalmonellaLabelValue)diseaseLabel.g
        CattleSalmonellaLabelValue deltaValue = (CattleSalmonellaLabelValue)diseaseLabel.getDeltaVa
        deltaValue.reset();

        // Add code here to compute and set the delta (in deltaValue) from the current state (in cu
        // ...

        // If you need to take into account any interventions currently in place for the region, us
        // code snippet as a sample:

        // StandardInterventionLabel sil = findInterventionLabel((Node)diseaseLabel.getIdentifiable
        // if(sil != null) {
        //     double vaccinations = ((StandardInterventionLabelValue)sil.getCurrentValue()).getVaccin
        //     ... add code to include vaccinations (isolations etc.) into your delta calculation he
```

# STEM Visual Editor



# Expression Language



- Domain-specific grammar for scientists
- Used to define transition rates between compartments
- “Hide the complexity” of STEM’s graph and data APIs
  - Bridge gap between literature and technology
- “Just-In-Time”: Interpret, compile, and inject “hot” code into running JVM
  - Integrates with Metamodel, Code Generator



- Visual Editor connects Model Generator and Expression Language in usable way
- Powered by Eclipse Graphical Editing Framework (GEF)
- Diagram Editor: adding, editing, removing compartments and transitions
- Expression Editor: specify conditions in Expression language
  - Code assistance
  - User documentation with NLS (Eclipse Babel)

# Generated Code is Fully Transparent

A screenshot of an IDE window titled "STEM". The main editor shows a Java file named "DemoFluExpressions.java". The code is as follows:

```
/**
 * Computes delta for transition s -> e
 * @generated
 */
protected double s_e (double t, long timeDelta, STETime time, DemoFlu model, DemoFluLabel label, DemoFluLabelValue labelValue, Node node) {
    double modulation=(1.0)
    /((1.0)
    +(Math.exp(
        (0.2)
        *(((CTDLFunctions.temperature(
            node
            ,
            time
            ,
            timeDelta
            ,
            t
            ))
        )-(10.0)
        ))
    ))
    );
    return (model.getTransmissionRate())
    *(labelValue.getS())
    *(org.eclipse.stem.diseasemodels.functions.CTDLFunctions.computeEffective(
        (org.eclipse.stem.diseasemodels.standard.StandardPackage.eINSTANCE.getSILabelValue_I())
        ,
        model
        ,
        node
        ,
        label
        ))
    );
    *(modulation)
    );
}
/**
 * Computes delta for transition e -> i
 * @generated
 */
```

The IDE interface includes a Project Explorer on the left, an Outline on the right, and a toolbar at the bottom with various tool icons like Scenarios, Models, Graphs, Triggers, Predicates, Sequencers, Properties, Tasks, Error Log, and Decorators.



- Reduced Lines of Code
  - 1000 LOC -> 5 LOC for simple SEIR model
- Improved Performance by generating optimal performance code
  - Temporal Caching
  - Appropriate Data Structures
  - Eliminating primitive boxing/unboxing
- Correctness
  - Visual Editor makes compartments, transitions obvious
  - User can define meaning of parameters, ... right at the beginning

# Generated Plug-in is hot injected and available at runtime



The screenshot displays the STEM software interface. The top menu bar includes 'STEM', 'File', 'Tools', 'Edit', 'New', 'Search', 'Run', 'Window', and 'Help'. The status bar shows 'STEM', 'New Solver', and system information: '(Charged) Oct 22 8:31 AM James Kaufman'.

The main workspace is divided into several panels:

- Project Explorer:** Shows a tree view of the project structure, including folders like 'AsiaMosquitoDemo', 'AutomatedExperimentExample', 'Demo', 'GlobalEarthScience', and 'org.my.prefix.demomodel'. The 'GermanyDemo.scenario Scenario' is selected.
- Graph Map:** Displays a map of Germany with a red overlay. The title is 'GermanyDemo | Linear Scale | Gain x1.0'. Below the map, there are controls for 'Now Displaying' (set to 'flu.standard Demo Flu'), 'Edges', and 'Labels Colors Mapping'.
- Simulation Control:** Shows the simulation progress: '[0] GermanyDemo', 'Progress : 25%', 'Status : Paused', and '[459] Time : Mon Sep 01 12:00:00 PDT 2014'. It includes play, pause, and stop buttons.
- Time Series:** Two plots are shown for 'DRESDEN, 10, DR DE-DR' and 'BERLIN, 03, BE DE-BE'. Each plot shows variables over time (0 to 534 days). The legend includes 'Disease Deaths', 'E', 'Incidence', 'I', 'R', 'Population Count', and 'S'. The 'Now Displaying' section for each plot shows 'flu.standard Dem' and 'human' with a 'remove' button.



# Thank you for your attention

## Matthias Filter and the BfR STEM team

Alexander Falenski  
Christian Thöns  
Armin Weiser  
Jan-Frederik Wigger  
Taras Günther  
Annemarie Käsbohrer  
Bernd Appel

Federal Institute for Risk Assessment  
Max-Dohrn-Str. 8-10  
D-10589 Berlin  
Tel. +49 30 - 184 12 - 0  
Fax +49 30 - 184 12 - 47 41  
[bfr@bfr.bund.de](mailto:bfr@bfr.bund.de)  
[www.bfr.bund.de](http://www.bfr.bund.de)