Building User Interfaces for Models

Nathaniel Osgood
MIT 15.879

April 25, 2012
Lecture Focus: Creating Custom User Interfaces using "Controls"

• 'Controls' are "widgets" that allow for obtaining user input
  – These widgets have properties that can be set at both design and run (execution, simulation) time

• By setting the properties of these controls at design time, we can
  – Establish their general logical & visual properties
  – Establish their correspondence with model variables

• These controls can be used by the user during simulation to set assumptions in the model
Hands on Model Use Ahead

Load Previously Built Model: MinimalistSIRNetworkABM

After change, suggest saving as “UISupportedMinimalistSIRNetworkABM”
Recall: Hardcoded Exposure Rate

“Hard-coded” rate
Add a Related Parameter to Main
Setting the Transition to Refer to the Parameter in Main

Note that "exposureHazard" lives in "Main". To get a reference to the "Main" object, we call "get_Main()" on ourselves.
Resulting Expression

```
pathogenExposure - Transition

<table>
<thead>
<tr>
<th>Name: pathogenExposure</th>
<th>Show name</th>
<th>Ignore</th>
<th>Show at runtime</th>
</tr>
</thead>
</table>

Triggered by: Rate
Rate: this.get_main().exposureHazard

Action:
this.send("Infection", RANDOM_CONNECTED);

Guard: 
```
Reminder: An Explicitly Specified Population Size
A Parameter Giving the Population Size
Setting the Population to Use the Parameter Value
Reminder: The Existing Experiment
Minimalist Network ABM Model

Experiment setup page

Run the model and switch to Main view
Reminder: Pushing the Button Shows the Simulation Visualization
Understanding the Button’s Actions

MinimalistNetworkABMMModel

Experiment setup page

Run the model and switch to Main view

button - Button

General

Name: button

Label: Run the model and switch to Main view

Enabled: state() == IDLE

Action:

run();
getEngine().getPresentation().setPresentable( getEngine().getRoot() );
Adding a Slider to Represent the Population Size

Fill in this information
Setting the Simulation Parameter Values to Use the Slider Setting

The value of the slider: \( \text{sliderPopulationSize}.\text{getIntValue()} \)
Choosing a High Value on the Slider
Resulting Network – Large Population
Choosing a Low Value on the Slider
Resulting Network -- Small Population
Adding (Static) Text Labeling Slider

Fixed text – doesn’t change over time
Creating a Text Element to Give the Slider Value

This text is initially blank, but we’ll set it elsewhere to change over time with the slider value.
Dynamic Properties to Report the Slider Value

By setting this expression to determine the dynamic value of the text field “Text” property, the string associated with this text will automatically change with the slider value.
Example Resulting Output

**MinimalistNetworkABMMModel**

*Experiment setup page*

Run the model and switch to Main view

Population size: 723
Reflecting on Temporal Specificity of UI Elements

• The user interface component (slider) we created thus has had its value used to set the initial state of the model (the population size)

• User interface components can also be used to vary assumptions dynamically during runtime
  – For example, vary parameter values
Example: Creating a Slider to Dynamically Vary the Infection Hazard

Set these as the upper and lower bounds of the slider.

Link so that changing the slider automatically changes the exposureHazard parameter.
A High Slider Value Leads to a More Rapid Spread
Dropping the Slider Value (Exposure Hazard) to 0 Can Stop the Spread
Recall: The Initial Infection Seed

This delivered an infection message to a randomly selected person in the population.
Cut Text from Startup Code for Main
Setting the Button to Seed a New Infection

This is the action the button will perform when pushed:

```java
environment.deliverToRandom("Infection");
```
With Multiple Presses, Multiple “Seed” Infections
Add a Contingent Reporting Variable

Initial value should be false
Contingent Infection Reporting

This makes the reporting contingent on the value of isReportingEnabled

```java
if (this.get_Has()).isReportingEnabled)
    tracein(this + "has been infected!")
```
Contingent Recovery Reporting

This makes the reporting contingent on the value of `isReportingEnabled`
Enabling Reporting

Link to the "isReportingEnabled" parameter
Unless Reporting is Enabled (i.e. Checkbox is Checked), No Output
Enabling Reporting Allows Output
Cleaning Up by Separating the Network Display Space from Other Model Components

This is the display “origin” for the agents. Positive coordinates for the agents will yield locations visually to the right and below this.
Hands on Model Use Ahead

Load Example Model:
HardcodedMinimalistNetworkABMMo
delWithFileDrivenNetworkStructure
Recall: “Hardcoded” File Names

This currently “hardcodes” that we are opening a particular Pajek file.

```java
// establishNetworkTransitionsAndPopulationsFromConnectivityMatrixFile("C:\Users\Classes\";
establishNetworkTransitionsAndPopulationsFromPajekNetworkFile("C:\Users\Classes\15879.sp environment\applied\layout()"); // now that established connectivity, perform layout
```
Creating a Parameter to Communicate the Network File Name & Location ("Path")

Indicate that this parameter holds a (reference to a) String.
Creating an “Enum” to Encode the Possible Types of the Specified File

Specifies legal types of files

```
enum NetworkFileType { Pajek, ConnectivityMatrix }
```
Creating a Parameter to Encode the Network File Type

Specifies that this parameter encodes the legal types of files (as specified by the NetworkFileType enumeration)
HardcodedMinimalistNetworkABMModelWithFileDrivenNetworkStructure - Model

Experiment setup page

Run the model and switch to Main view
Referring to the External Java Swing Library

Choose “Add” to add the reference to the “swing-layout-1.0.1.jar” file, which contains the Java library containing the “JFileChooser” control (dialog box). [This is freely downloadable; some other versions may also do]
Adding a Reference to the Java “Swing” File Chooser Component

We add a reference here to the “JFileChooser” control, since we need to use it.

```java
import javax.swing.JFileChooser;
```
Adding a Button “buttonSelectFile”
Add an EditBox
editboxNetworkFilePathAndName
Adding a Label for the Filename

This is static text
Logic to Set the File Name

Here, we open the Dialog box, which returns a value indicating the result.

If this return value indicates that the user wants to go ahead with the chosen file...

...set the text in the Editbox to The “fully Qualified” filename (including path)
Add these values. Note that the order is important – it must be in the same order as in the NetworkFileType Enum (since we’ll just use the position to select the appropriate enum value).
For the network file path & name parameter, we just take the value from the editbox.

For the network file type parameter (which requires an enum value), we just pass the enum value at the index given by the “Radio button” (the first enum value [i.e. at index 0] if the first radio button is selected, the second if the second radio button is selected).
For the startup code for Main, we call the appropriate method to process the specified file, where the identity of that method is indicated by the specified NetworkFileType