

Events in AnyLogic

Nathaniel Osgood

Using Modeling to Prepare for Changing Healthcare
Needs

Duke-NUS

April 16, 2014

Events & Scheduling in AnyLogic

- Reminder: In simulating stock & flow models, time advances in steps
 - Euler integration: Fixed-sized Steps
 - Runga-Kutta: Fixed or variable sized steps
 - For each timestep, we compute the flows & update the stocks
- AnyLogic jumps from “event” to “event”
 - The data structure that keeps track of such events is called the “schedule”
 - The associated process is called the “scheduler”

Implicit Events we've Seen

- Transitions
 - Fixed rate (Poisson arrival)
 - Timeout
 - Condition
 - Message transmission (schedules event for the receiver)
- Starting a model
- Stopping a model
- In this course, we term these *implicit events* because they are not reified as objects in the model
- To handle these events, code is inserted into certain handler areas for each of different sorts of classes

Example: Built-In Events (Agent 1)

The screenshot displays the AnyLogic Professional interface. The main workspace shows a diagram with a 'Person' agent and a 'connections' link. A red text box with a red arrow points from the text to the 'Message type' field in the Properties panel.

“Handler”: Code is executed when the specified event (e.g., arrival at a destination, message arrival) occurs.

Properties - Link to agents

- Collection of links
- Single link

This standard link is always bidirectional

Communication

These actions are executed for messages from all connections

Message type:

On message received:

```
infectionStatechart.receiveMessage(msg);
```

Forward message to:

Statecharts	
<input type="checkbox"/> infectionStatechart	
<input checked="" type="checkbox"/> movementStatechart	

Time units: days

Example: Built-In Events (Agent 2)

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a diagram with a central node labeled 'Person' connected to other elements like 'city', 'main', and 'connections'. The 'Person' node is highlighted in the 'Projects' pane on the left.

The 'Person - Agent Type' configuration panel is visible at the bottom, showing the 'Agent actions' section. The 'On startup:' and 'On destroy:' fields are highlighted in red, indicating they are the focus of the example. The 'On arrival to target location:', 'On before step:', and 'On step:' fields are also visible but not highlighted.

The 'Entity actions' section is partially visible at the bottom of the configuration panel.

The status bar at the bottom right indicates 'Time units: days'.

Example: Built-In Events (Main)

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a grid with a hand icon in a circle on the left and two text boxes on the right. The top box is titled "Initial Parameters Of Simulation" and contains the text "Initial cities Size" and "Mean Recovery Time". The bottom box is titled "Color Of Humans" and contains three colored icons: a red person icon labeled "Infective", a green person icon labeled "Susceptible", and a grey person icon labeled "Rec".

The left sidebar shows a project tree with the following structure:

- HybridABMNetworkModelingUsing/
 - HCWorker
 - Main
 - Person
 - Simulation: Main
- HierarchicalCityPopulationModelWith/ul>- City
- Main
- Person
- HierarchicalCityPopulationMode
- HierarchicalCityPopulationMode
- HierarchicalCityPopulationMode

- Extended SIR Agent Based Calibration/ul>- Main
- Person
- TestSimulation: Main
- Calibration: Main
- CalibrationMystery: Main
- MonteCarlo2DHistogram: Main
- ChildhoodInfectiousDiseaseModelUsing/ul>- Main
- Person
- ChildhoodInfectiousDiseaseMod
- ChildhoodDiabetesExploratoryModelUsing/ul>- Main
- Person
- ChildhoodDiabetesExploratoryM
- GriddedSystemDynamicsUsingAnylo/ul>- Main
- Patch
- Simulation: Main



The bottom panel shows the "Main - Agent Type" properties. Under "Agent actions", the "On startup" event is highlighted in red and contains the code: `deliverToRandomAgentInside ("Infection");`. Other events like "On destroy:", "On arrival to target location:", "On before step:", and "On step:" are listed but empty.

The bottom status bar shows "Time units: days" and "X=...36".

The Schedule

- At a given time, the schedule keeps track of a number of queued events
- Events may get added to the schedule (e.g. when we enter a new state)
- Events get deleted from the schedule
 - When they fire off and are complete
 - When another mutually exclusive event preempts them (e.g. a person dies before they recover from an infection)

Explicit Events

- Explicit Events can also be declared from the palette  Event  Dynamic Event
 - Dynamic events can have multiple instances
 - Each instance can be scheduled at different times
 - The instances disappear after event firing
 - Regular (static) events can be rescheduled, enabled/disabled, but can only have one scheduled firing at a time
- There are some subtleties with explicit events

(Explicit) Event Subtleties

- Be very careful of what you count on for recomputation of rate – may think was recomputed, but hasn't been
- Event rates (and likely event timeout times) are only computed occasionally, not continuously
 - These are computed when
 - Explicitly call event methods
 - `start()`
 - `restart()`
 - `onChange()`
 - » e.g. if wish to update rates associated with transitions, *Main* can periodically call *onChange()* on each agent
 - An event in Main can take care of this task
 - When event fires and requires restarting
 - (For outgoing transitions) when enter a state in a statechart
- Calling “reset” will disable a rate until re-enable (e.g. with call to *restart()*)

Event Times: Common Options for Event Scheduling

- At a specified rate (Poisson arrivals)
 - Interarrival time is exponentially distributed!
 - Mean time between events is reciprocal of rate (i.e. $1/\text{rate}$)
- One-time
 - Can go off at a particular time (specified as a calendar time or as a double-precision value)
- At some initial time and then cyclically beyond with set “timeout” period
 - The timeout period is set according to the time unit
 - This goes off after *exactly* the timeout time
- When boolean condition changes (depends on *onChange* being called)
- Manually (via `restart()`) – see following slides)

Event Subtleties

- Be very careful of what you count on for recomputation of rates – may think was recomputed, but hasn't been
- Event rates (and likely event timeout times) are only computed occasionally, not continuously
 - These are computed when
 - Explicitly call event methods
 - start()
 - restart()
 - onChange()
 - When event fires and requires restarting
 - (For outgoing transitions) when enter a state in a statechart
- Calling “reset” will disable a rate until re-enable (e.g. with call to *restart()*)

Dynamic Events (Closure-Like)

- Like a static event, a dynamic event is associated with an *action* to invoke when it occurs
- A *static* event has a single associated schedule
- Just as a class can be associated with multiple instances, Dynamic events can have *multiple instances*
 - Each instance can be scheduled at different times
 - The schedule for each different instance proceed in parallel
 - The instances disappear after event firing
 - We can think of each dynamic event instance as its own one-time (“one-shot”) event
- Schedule a dynamic event with `create_event(timeout, parameters...)`
 - The event will be “awoken” time *timeout* from now!

Parameterization of Dynamic Events

- With a dynamic event, we create the event during simulation, but at a different time than it occurs
- Frequently the action we want to performed in a dynamic event depends on specific context known at the time that it was created
 - For example, we want to create or delete a particular person, or a person with particular characteristics
- Specification of dynamic events at design time defines custom ‘parameters’ (‘arguments’)
 - Parameters values can be used to communicate context from time of creation of the dynamic event until when it fires
 - Particular values for these parameters are then given at time when dynamic event instance is created

Specifying a Dynamic Event Step 1

The screenshot displays the AnyLogic Professional software interface. The main canvas shows a model with a variable 'population' and a dynamic event 'ScheduleDeerBirth'. The 'Palette' window on the right contains various model elements, with 'Dynamic Event' highlighted. The 'Properties' window at the bottom shows the configuration for the 'ScheduleDeerBirth' event, including its name, visibility, and action.

1) Click here,

2) use mouse to click in Canvas to add Dynamic event

Click on the "Model" label in the "Palette" window

Specifying a Dynamic Event Step 2

The screenshot displays the AnyLogic Professional interface. The main workspace shows a dynamic event named 'ScheduleDeerBirth' connected to a 'population' variable. The Properties panel is open, showing the configuration for this event.

Properties Panel: ScheduleDeerBirth - Dynamic Event

Name: Show name Ignore

Visible: yes

Parameters

Name	Type
isFemale	boolean
genotype	int
latitude	double
longitude	double

Action

Time units: days

Attractive Use of Dynamic Events 1

Scheduling Future Birth at time of Conception

- Mating of deer during rut occurs long before births of fawns
- Contacts between deer during rut could be simulated in the model
 - At time of contact, create single dynamic event to schedule associated future birth
 - Could save away information of history relevance e.g.
 - Characteristics of parents
 - Infection status
 - Genotype
 - Stress level
 - Location of where conception occurred

Attractive Use of Dynamic Events 2

Adding in Individuals to Population over a Time Interval

- Dynamic events can be very handy if have a known number of actions that need to take place spread out over some period of time
- Example: Given: Known count of Immigrants with particular characteristics to be added to model population over course of each month
 - Suppose we don't know when these individuals arrive during the month
 - We can simply create the same count of dynamic events, whether each dynamic event takes care of
 - Creating a person with known characteristics
 - Adding that person to the model population

This approach will be discussed in an upcoming guest lecture