How to Use the Broadwick Framework to write a simple model

Quick Start Notes

S. J. Lycett

19 June 2014

University of Glasgow | Institute of Biodiversity, Animal Health & Comparative Medicine

epic Centre of Expertise on Animal Disease Outbreaks

To do list

- Decide on parameter names and create a configuration XML file
- Write specific classes for your model, implementing interfaces:
  - Amount Manager
  - Simulator Controller
  - Simulation State
- Extend the Observer class to make suitable model output files
- Extend the Model class
  - Override init, run, and finalise methods

BroadwickExamples

- See simple examples at :
  https://github.com/EPICScotland/BroadwickExamples

Note:
Netbeans
directory
structure

- Package epic.broadwickExamples
  – Contains BlankModel (just writes to screen) and DummyModel (reads parameters from XML and writes to screen only).
- Package epic.basic, BasicSIRModel
  – Stochastic SIR compartmental model, outputs numbers of S, I, and R over time
- Package epic.sir, IndividualSIRModel
  – Individual based SIR type stochastic compartmental model, also outputs who infected whom
- Package epic.network, NetworkSIRModel
  – extends the individual stochastic SIR model above to only allow infections over a network (example network included)

Model Init Method

- This method initialises the Model within the Broadwick framework
- It should include reading the parameters from XML configuration file
- The objects below need to be initialised, either within the init method (as in the examples here), or at the start of the run method (depending on other requirements such as multi-threading)
  - Initialise Transition Kernel
  - Initialise Amount Manager
  - Initialise Simulator (simulation engine) with Transition Kernel and Amount Manger
  - Initialise Controller (stops simulations with conditions)
  - Initialise Observer (generates output files)
  - Register Observer with Simulator

Model Run Method

- Actually runs the simulation by performing events
- The Stochastic Simulator classes provided in Broadwick already provide a run method:
  - The Gillespie Algorithm
  - The Fixed Step Tau Leap Algorithm

- The run method implemented in the examples is just this:

```
@Override
public void run() {
        simulator.run();
}
```

Model Finalise Method

- This method is called at the end of the simulation
- You might like to make a final model state output

- Or maybe just write a message to the log file:

```
@Override
public void finalise() {
        log.info(“Final simulation time ="+simulator.getCurrentTime());
        log.info(“Final model state ="+amountManager.toVerboseString());
}
```

Summary of Classes

Name	Description
Transition Kernel	Contains simulation events to be performed, e.g. one individual becomes infected.
Amount Manager	This is the class which implements the epidemiological model. It has the performEvent method, which updates the model state and generates the next set of simulation events which are added to the Transition Kernel.
Stochastic Simulator	Chooses which of the events will be performed next from the Transition Kernel, and requests the Amount Manager to actually perform the chosen event.
Observer	Has methods to record the model state just before and after an event is performed – typically this class generates the output files.
Controller	Has the goOn method, which returns true to continue with the next step of the simulation, or false to terminate. For example the simulation is stopped if a maximum time is reached.